# *Creación de una guía práctica de traducción automática neuronal para estudiantes de traducción*

**Nombre:** Clara García Abiétar

**Línea de investigación:** Tecnologías de la traducción

**Tutor:** Mikel L. Forcada Zubizarreta

**Fecha:** 05/06/2019

# Trabajo de

# Fin de Grado de

# Traducción e Interpretación

Clara García Abiétar

cga79@alu.ua.es

# Índice

**RESUMEN**

Últimamente ha aumentado en el interés por un nuevo paradigma de traducción automática (TA): la TA neuronal (TAN). La TAN está empezando a desplazar a su predecesora basada en corpus, la TA estadística (TAE). Aunque existen varios paquetes de software abierto para realizar la TAN, su instalación, su entrenamiento y su uso no son sencillos para jóvenes profesionales de la traducción. En este trabajo, he creado independientemente en mi ordenador personal un sistema de TAN, lo he entrenado para una tarea francés–español, y lo he usado en una tarea realista. Uno de los resultados es una breve guía (*how-to*) en la que se describe el proceso seguido, escrita de forma adecuada para estudiantes de último curso del grado de traducción.

**Palabras clave:** traducción automática neuronal, guía, instalación, entrenamiento, aplicación.

**RÉSUMÉ**

« Création d'un guide pratique de traduction automatique neuronale pour les étudiants en traduction »

Récemment, l'intérêt pour un nouveau paradigme de la traduction automatique (TA) s'est accru : la TA neuronale (TAN). La TAN commence à supplanter sa prédécesseure à base de corpus, la TA statistique (TAS).0 Bien qu'il existe plusieurs paquets logiciels libres pour réaliser la TAN, leur installation, leur entrainement et leur utilisation ne sont pas faciles pour les jeunes professionnels de la traduction. Dans ce projet, j'ai créé de manière indépendante sur mon ordinateur personnel un système de TAN. Je l'ai formé pour une tâche franco–espagnole et je l'ai utilisé dans une tâche réaliste. L'un des résultats est un bref guide (*how-to*), dans lequel le processus suivi est décrit, rédigé d'une manière adaptée aux étudiants en dernière année de la licence de traduction.

**Mots-clés :** traduction automatique neuronale, guide, installation, entrainement, application.

NOTA

A lo largo de este trabajo de fin de grado algunos términos se presentan acompañados del símbolo de un libro (🕮); esto significa que sus definiciones se pueden consultar en el Apéndice II – Glosario.

## 1. Introducción

*Traducir es hacer un viaje por un país extranjero*

George Steiner

El presente trabajo de investigación constituye el trabajo final de grado de Traducción e interpretación. La idea fue proporcionada por Mikel L. Forcada, tutor del trabajo, y en principio se basaba en el entrenamiento de un sistema de *traducción automática neuronal* 🕮. A raíz de esta se fue desarrollando la posibilidad de ir más allá: además de aprender cómo funciona el entrenamiento de un sistema de TAN creado por una persona ajena, crear una guía para que otro estudiante pudiera hacer el mismo experimento.

La traducción automática se engloba dentro de la disciplina de la lingüística computacional, «rama de la lingüística en la que se emplean conceptos computaciones para la elucidación de problemas lingüísticos y fonéticos» define Crystal (2000: 345); y se define como un proceso de la traducción en la que un sistema informático compuesto por un ordenador y unos programas específicos produce un texto en la lengua meta que puede ser considerado una traducción en bruto de un texto informatizado en la lengua origen. Hasta el momento se han desarrollado dos tipos de TA: la *traducción automática basada en reglas* 🕮 y la *traducción automática basada en corpus* 🕮. También se podría hablar de la existencia de un tercer grupo de TA: el resultado de la hibridación de estos dos últimos. El objeto de

mi trabajo, la traducción automática neuronal, forma parte del segundo grupo. La TAN empezó a desplazar a su predecesora, la *TA estadística*, 🕮 alrededor de 2015 y, aunque la idea ya había surgido en los años 90 por Castaño & Casacuberta (1997) y Forcada & Ñeco (1997: 453–464), no volvió a retomarse hasta el año 2013 debido fundamentalmente a las limitaciones de hardware que había en aquella década. La TAN también recibe el nombre de traducción basada en *aprendizaje profundo* 🕮 (en inglés, *deep learning*) debido a las numerosas *capas de neuronas* 🕮 que trabajan en redes neuronales artificiales.

### *1.1. Objetivo del trabajo*

Mi trabajo se dividió en cuatro partes esenciales:

I. La documentación a partir de lecturas recomendadas por mi tutor para introducirme en la materia;

II. La realización de pequeños experimentos entrenando un sistema de traducción automática neuronal de vietnamita–inglés que formaba parte de una guía de instalación de un software de TAN;

III. La creación de un sistema de traducción automática neuronal desde cero y el posterior entrenamiento de este habiendo seleccionado un pequeño corpus francés–español de temática periodística para poder traducir textos originales en francés a la lengua española y así utilizar este sistema resultante para una tarea de traducción realista en la misma temática;

IV. La preparación de una guía que describa todo este proceso, escrita de forma adecuada para que estudiantes de último curso del grado de traducción también puedan llevar a cabo este experimento, y que contenga dos apéndices imprescindibles (un glosario con las palabras que me costó entender y de las que no había mucha información en Internet en español y una lista de comandos para los *intérpretes de órdenes* 🕮 *Bash* 🕮 y de *Windows* 🕮).

El objetivo final fue el de contribuir a dejar atrás esos estereotipos sobre la escasez de conocimiento y habilidades sobre temas informáticos de las personas que estudian carreras de letras, y animar a cualquier estudiante de traducción a experimentar y aprender a liberarse de estos miedos infundados. También se eligió hacer este tipo de trabajo por la escasez de artículos en español. Por lo tanto, una guía que estuviera lo más completa posible sobre la traducción automática neuronal y que estuviera redactada en español sería una buena opción para fomentar el uso del español y cuestionar la universalidad del inglés. El objetivo del trabajo fue ambicioso teniendo en cuenta el poco tiempo del que se disponía y el tiempo que se necesitaba para entrenar un sistema de traducción automática neuronal con ciertas limitaciones de hardware. Claramente este sistema de TAN no podía tener la misma envergadura que un traductor automático diseñado y entrenado por profesionales como lo serían DeepL o el sistema de traducción automática más conocido mundialmente, el traductor de Google, los cuales se basan en corpus muy amplios con miles e incluso millones de pares de frases traducidas en gran variedad de lenguas y grandes instalaciones con ordenadores especializados de gran potencia. De hecho, el primer uso comercial de la TAN fue en 2016 por parte del traductor de Google, que hasta ese momento había sido un traductor automático estadístico. A pesar de esto, se han obtenido resultados muy satisfactorios y con mayor calidad de lo que yo esperaba, y lo que es mejor, motivadores para seguir después del grado aprendiendo sobre la traducción automática neuronal. Cuando el tutor me propuso la idea me pareció todo un reto realizar por mi cuenta debido a los miedos y estereotipos mencionados anteriormente, pero con su guía y ayuda se ha podido realizar el trabajo y sobrepasar los obstáculos. Este proyecto puede mejorarse en algunos ámbitos, como por ejemplo, profundizar en el aprendizaje del lenguaje *Python*,[1] 🕮 puesto que tengo conocimientos muy básicos; desarrollar una memo-

[1] Se requiere Python para poder ejecutar Tensorflow. Python es un lenguaje que se aprende fácil, es popular y contiene muchos programas escritos en él útiles para entrenar los sistemas de TAN.

ria de traducción con traducciones en bruto realizadas por el sistema francés–español de TAN o mejorar el indicador automático de calidad (*BLEU* 🕮) de la traducción en bruto del sistema para disminuir el tiempo de postedición del traductor, entre otros. Como se puede ver, el aspecto más positivo del trabajo es que se trata de un experimento que renueva constantemente los objetivos, es decir, no tiene porqué acabar aquí, sino que se puede continuar más allá de este trabajo de fin de grado.

### *1.2. Importancia de la TAN en los últimos años*

Desde 2015 la traducción automática neuronal ha ido creando más y más interés en los profesionales del ámbito de las tecnologías de la traducción, puesto que sus resultados son prometedores. «En una época en la que el volumen de demanda de traducciones es superior a la capacidad de los traductores» Hutchins & Somers (1995: 26), el uso de un sistema de TA supone un apoyo importante para los traductores y, por consiguiente, una razón para incentivar la mejora y perfeccionamiento de estos sistemas. Este tipo de traducción automática está desplazando actualmente a su predecesora basada en corpus, la traducción automática estadística. Los sistemas de TAN que han participado en los concursos internacionales WMT en the Conference on Machine Translation[2] han resultado producir los mejores resultados en lo que se refiere a valoraciones subjetivas y a las medidas de evaluación automáticas que las comparan aproximadamente con las traducciones preexistentes, como por ejemplo, *BLEU* 🕮, que aproximan la mejora de su impacto real en la productividad de los traductores. Forcada (2017: 304) explica una situación en la que el sistema de TAN obtiene mejores resultados que el de traducción automática estadística (TAE):

---

[2] Por ejemplo, la edición de 2018 de WMT (http://statmt.org/wmt18/).

> Adequate translations usually require placing the equivalents of source words in a completely different order. Automatic analysis finds that NMT (Toral and Sánchez-Cartagena 2017; Bentivogli et al. 2016) produces reorderings that resemble more those of reference sentences than those produced by SMT (the latter paper reports a 50% decrease in "word order errors"). In an English–German task, Bentivogli et al. (2016) also found that NMT produces "less morphology errors (–19%) [and] less lexical errors (–17%)" than SMT.[3]

He escuchado opiniones de personas tanto del grado como de fuera del círculo de la traducción que se negaban a aceptar que una máquina pudiera disminuir el tiempo de trabajo de un traductor; al contrario, afirman que le haría perder mucho tiempo posteditándolo. Ya fuera por miedo a la desaparición de multitud de puestos de trabajo, ya fuera por la obstinación de que las máquinas no pueden ser tan inteligentes. Como concluye en uno de sus artículos Forcada (2015: 222):

> One should never expect the MT system – however *good* – to understand the text, to always solve ambiguities properly and to produce texts conforming to the TL norms or fit for the intended purpose of the translation.[4]

La TA no quitará puestos de trabajo a traductores profesionales, sino que agiliza su proceso de traducción: para disminuir el tiempo de postedición de una traducción en bruto, es necesario un sistema de traducción automática eficaz y eficiente y que ha sido entrenado con corpus bilingües formados por millones de pares de frases alineadas, idealmente de temática relacionada. Para ello, cuántos más textos haya disponibles en Internet o en otro sitio, traducidos por profesiona-

---

[3] «Las traducciones adecuadas normalmente requieren la colocación de los equivalentes de las palabras de origen en un orden completamente diferente. El análisis automático encuentra que la TAN (Toral y Sánchez–Cartagena 2017; Bentivogli et al. 2016) produce reordenamientos que se asemejan más a los de las oraciones de referencia que a los producidos por la TAE, traducción automática estadística (este último artículo reporta una disminución del 50% en los "errores en el orden de las palabras"). En una tarea inglés–alemana, Bentivogli et al. (2016) también hallaron que la TAN produce "menos errores morfológicos (-19%) [y] menos errores léxicos" (-17%)" que la TAE».

[4] «Nunca se debería esperar que el sistema de traducción automática, por muy *bueno* que fuera, comprendiese el texto, resolviera siempre correctamente las ambigüedades y produjera textos que se ajustasen a las normas de la LM o que se ajustasen al propósito previsto de la traducción».

les, y cuántas más traducciones fiables de estos haya, más material tendrán los sistemas de traducción para mejorar sus resultados. Este argumento se podría rebatir con el razonamiento de que llegará un momento en que se alcanzará un número considerable de traducciones disponibles en la red y que los sistemas no necesitarán más para entrenarse porque ya estarán lo suficientemente bien entrenados. Pero se estaría olvidando el hecho de que las lenguas están vivas y como todo ser vivo crecen, evolucionan. El más claro ejemplo es que, al necesitarse corpus tan grandes para el entrenamiento de los sistemas, hay muchas lenguas que no disponen de tal cantidad de corpus bilingües para entrenarse.[5]

### *1.3. Cómo funciona la TAN*

La *traducción automática neuronal* 🕮 utiliza *redes neuronales* 🕮, «las cuales más bien deberían llamarse redes neuronales artificiales», especifica Forcada (2017: 292), entrenadas sobre corpus paralelos de miles o millones de pares de frases paralelas. Antes de empezar el entrenamiento de un sistema, se deben crear varios corpus: el de entrenamiento, el de desarrollo y el de test. El de *entrenamiento* 🕮 es el corpus más grande de los tres, típicamente con centenares de miles o millones de pares de segmentos (oraciones, frases paralelas en dos idiomas) y, según indica su nombre, sirve al sistema para entrenarlo con sus pares de segmentos alineados. El de *desarrollo* 🕮 está compuesto por unos pocos millares de pares de segmentos que permiten detener el entrenamiento para evitar «sobreentrenarlo» y que el sistema no se sobreentrene, es decir, no aprenda de memoria las traducciones del conjunto de aprendizaje en tal grado que le impida generalizar a nuevas traducciones no vistas. Por último, el de *test*, 🕮 al igual que el de desarrollo, está formado por entre 1 000 y 3 000 pares de segmentos que no se usan en el

---

[5] Además, también está el razonamiento de la reescritura de los grandes clásicos: ¿quién podría leer *Don Quijote de la Mancha* en la versión original que escribió Miguel de Cervantes publicada en 1605 y comprender su historia? Quizá la famosa saga de Juego de Tronos dentro de unos siglos tendrá que ser reescrita.

entrenamiento y proporcionan una indicación independiente del rendimiento del sistema de TAN. Un corpus sobreentrenado no obtendría un rendimiento alto en el conjunto del test.

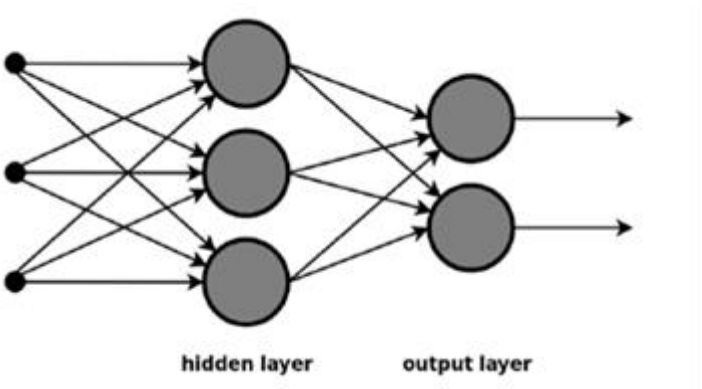
Como se ha mencionado anteriormente, la TAN se relaciona automáticamente el término a *aprendizaje profundo* 🕮. Según explica Forcada (2017: 294) en el aprendizaje profundo:

> Representations are usually deep (hence the buzzword *deep learning*): they are not built in one shot, but in stages from other shallower representations or layers. These layers usually contain hundreds of neural units: weights connect all units in one layer with all units in the next layer; the number of connections ranges in the thousands.[6]

Esto se debe a la estructura de los sistemas de traducción: trabajan con varias de *capas de neuronas* 🕮, es decir, de grupos de neuronas que únicamente reciben conexiones de neuronas de fuera de la capa y las envían a neuronas de fuera de la capa. Existen las capas de entrada, las de salida y las que están ocultas, que conectan las de entrada con las de salida. Cada capa de neuronas realiza una etapa de computación. En la siguiente imagen se muestra una red neuronal compuesta por tres capas de entrada que conectan con tres neuronas en una capa oculta y que a su vez se conectan con dos neuronas de salida.

---

[6] «Las representaciones suelen ser profundas (de ahí la palabra de moda, *aprendizaje profundo*): no se construyen de golpe, sino en etapas a partir de otras representaciones o capas más superficiales. Estas capas suelen contener cientos de unidades neuronales: los pesos conectan todas las unidades en una capa con todas las unidades en la siguiente capa; el número de conexiones es de miles».




La activación de cada neurona artificial de la red (su excitación o depresión) depende de la activación de las otras neuronas y del peso de las conexiones. Así, el signo y la magnitud del peso de la conexión determina el comportamiento de la red neuronal. Si el peso es positivo, las neuronas tienden a excitarse o deprimirse la una a la otra. Mientras que si el peso es negativo, las neuronas conectadas tienden a permanecer en el estado opuesto. Este efecto aumenta con la magnitud del peso. El entrenamiento consiste en fijar los pesos a los valores necesarios para así asegurar un comportamiento determinado. Los niveles de activación de ciertos grupos de neuronas (normalmente los de una *capa* 🕮) forman representaciones de la información que están procesando. En la traducción automática neuronal las palabras o caracteres, se procesan de forma paralela por parte de las neuronas de cada capa: los valores de activación de cada neurona constituyen representaciones de palabras y sus contextos. Estas representaciones (en inglés, *embeddings* 🕮) se aprenden automáticamente. Las activaciones de las neuronas de una *capa* 🕮 forman un *vector* 🕮 de valores, por ejemplo, entre - 1 y + 1, es decir, la primera neurona tiene activación + 0.4, - 0.89, + 0.26, 0, etc. Por ejemplo, en la siguiente imagen las palabras que comparten características semánticas parecidas en una capa con tres neuronas (ancho, alto y ancho) están más próximas las unas a las otras:

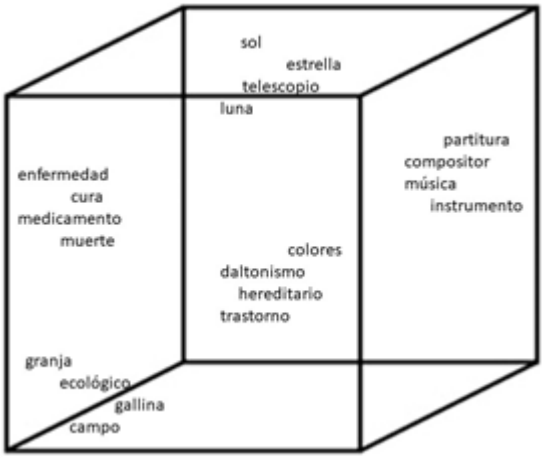



**Imagen inspirada en las transparencias de Felipe Sánchez para la asignatura de Tecnologías de la Traducción**

Si se tuvieran tres neuronas, cada una de las palabras de esta imagen sería un vector (x, y, z) donde *x*, *y*, *z* pueden tomar vectores entre, por ejemplo, – 1 y + 1. Por ejemplo, la palabra «sol» podría ser el vector (- 0,2, 0,1, 0,9).

Esta es la manera en la que el sistema de *TAN* 🕮 maneja las palabras. Las representaciones, aprendidas automáticamente por el sistema suele cumplir aproximadamente la aritmética semántica sin habérselo pedido expresamente al sistema. Así, la fórmula de c(«queen») – c(«woman») + c(«man») donde proporcionará un vector muy similar al de c(«king»), la misma conclusión a la que habría llegado un humano, Mikolov et al. (2013b). «c(x)» representa el *vector* 🕮 correspondiente a la palabra *x*.

### *1.3.1. Codificador y descodificador*

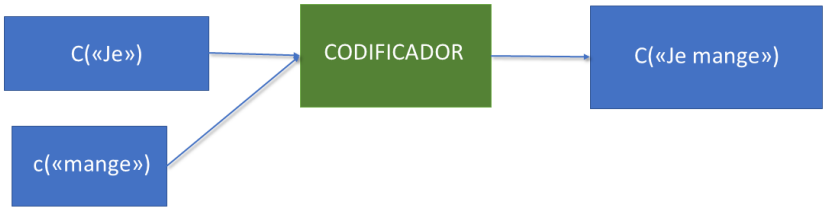
Una gran parte de los sistemas de TAN consiste en un *codificador* 🕮 y un *descodificador* 🕮, cada uno de los cuales es una red neuronal (recurrente) artificial especializada con una o varias capas. Por su parte, «el codificador analiza de izquierda a derecha y de derecha a izquierda la frase origen para generar una repre-

sentación vectorial de la misma», Casacuberta & Peris (2017: 69), a partir de representaciones de cada una de las palabras de origen.

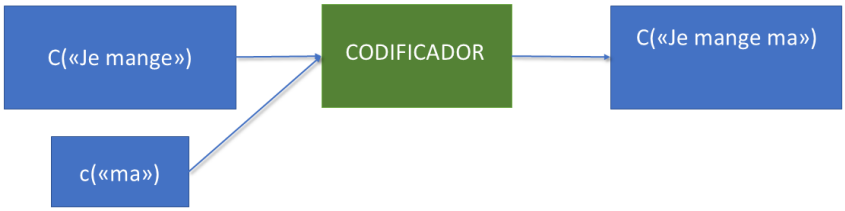
A través de la traducción al español de una frase en lengua francesa como «Je mange ma poire» se pueden explicar los pasos que realiza el codificador. El codificador construye una representación de esta oración que está constituida a su vez de representaciones de cada una de las palabras que la forman, c(«je»), c(«mange»), c(«ma») y c(«poire»). La codificación sigue los siguientes pasos:

I. El codificador combina una oración vacía representada como C(«») para indicar el inicio de la oración con la representación de la primera palabra c(«je») y produciendo la representación C(«Je»);

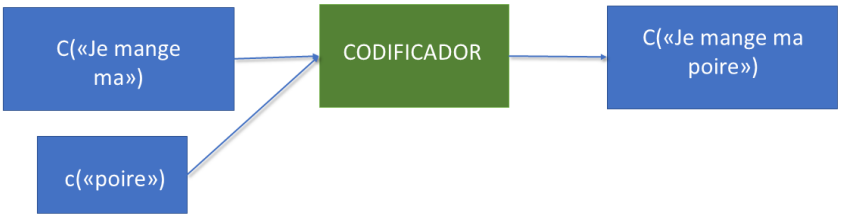

II. Continúa uniendo las representaciones de las palabras de la oración original hasta obtener la representación de la oración entera. Por lo que la siguiente sería: codificador [C(«Je»), c(«mange»)] = C(«Je mange»);



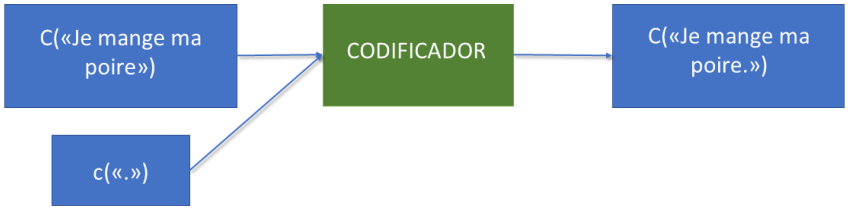

III. Continúa codificador [C(«Je mange»), c(«ma»)] = C(«Je mange ma»);




IV. El siguiente: codificador [C(«Je mange ma»), c(«poire»)] = C(«Je mange ma poire»);




V. Y, por último, codificador [C(«Je mange ma poire»), c(«.»)] = C(«Je mange ma poire.»).[7]




Después de este proceso, es el turno de la descodificación en la que, en un momento dado, «el descodificador genera la frase destino condicionado por la frase origen» por Casacuberta & Peris (2017: 69) prediciendo la palabra más probable en cada uno de los pasos dadas la frase original y las palabras meta anteriormente escritas. Para ello, usa el «estado» o «representación» que así llamamos

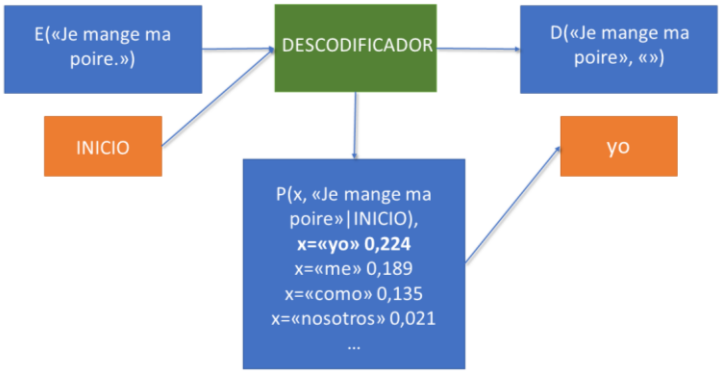
[7] Imágenes inspiradas en el artículo en *Translation Spaces 6:2* (2017) de Mikel L. Forcada.

D(a, b), la cual depende de *a* (la oración original) y *b* (la parte de la oración meta que ya ha quedado codificada). Se compara la forma de actuar del descodificador con la del predictor de texto de los teléfonos inteligentes, como indica Forcada (2017: 296):

> Most NMT systems are built and trained in such a way that they resemble a text completion device (analogous to the word prediction feature of smartphone keyboards).[8]

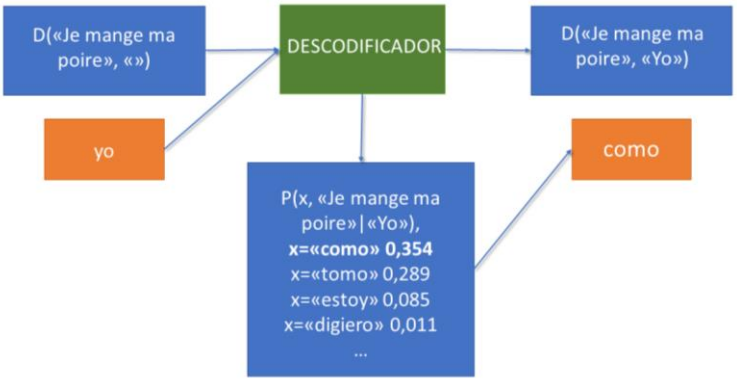
Con la oración anterior en francés ya codificada D(«Je mange ma poire.»), el descodificador elabora dos vectores: uno que represente el inicio de frase D(«Je mange ma poire», «») y otro que recoge todas las palabras posibles de la LM con su probabilidad en ese punto p(x|« Je mange ma poire», «»), siendo la notación p(x|«») la probabilidad de la palabra x dado «». El descodificador va avanzando palabra por palabra de la misma forma y va eligiendo la más probable en cada punto:

I. En primer lugar, D(«Je mange ma poire», «») con «Yo» producen D(«Je mange ma poire», «Yo») y p(x|«Je mange ma poire», «Yo»);

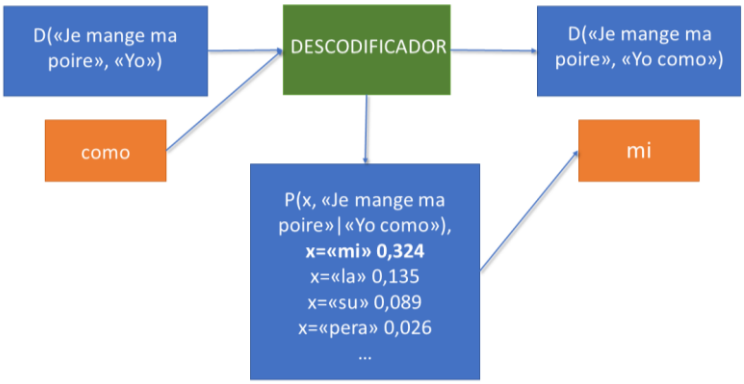



[8] «La mayoría de los sistemas de TAN están construidos y entrenados de tal manera que se asemejan a un dispositivo de predicción de texto (análogo a la función de predicción de palabras de los teclados de los teléfonos inteligentes)».

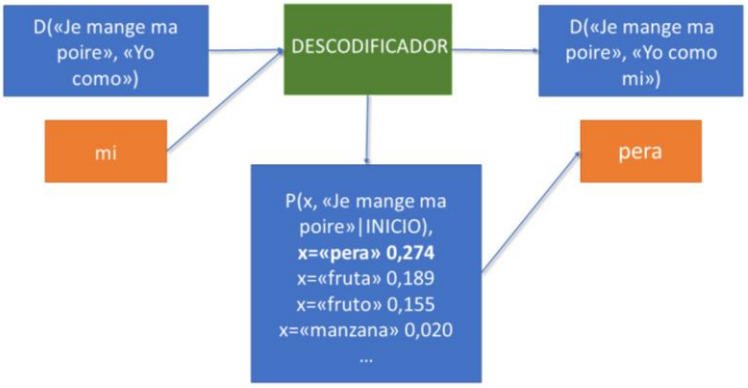
II. Continuando con D(«Je mange ma poire», «Yo») con «como» producen D(«Je mange ma poire», «Yo como») y p(x|« Je mange ma poire», «Yo como»);



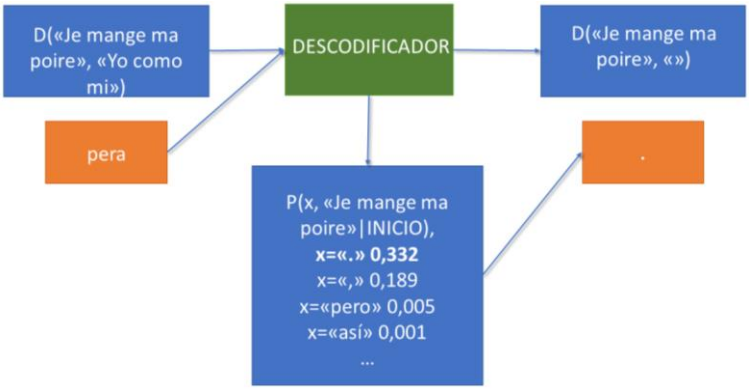

III. Descodificando D(«Je mange ma poire», «Yo como») combinado con «mi» producen D(«Je mange ma poire», «Yo como mi») y p(x|« Je mange ma poire», «Yo como mi»);




IV. Sigue D(«Je mange ma poire», «Yo como mi») combinado con «pera» producen D(«Je mange ma poire», «Yo como mi pera») y p(x|«Je mange ma poire», «Yo como mi pera»);




V. Y, por último, D(«Je mange ma poire.», «Yo como mi pera.») y p(x|«Je mange ma poire.», «Yo como mi pera.»). En este último ejemplo, la x más probable es un punto con lo que acaba la descodificación. Un sistema de traducción bien entrenado sabrá elegir la opción más adecuada para cada palabra porque será la que tendrá el mayor índice de probabilidad.[9]




## 2. Metodología

[9] Inicio del proceso de la descodificación. Inspirada en el artículo en *Translation Spaces 6:2* (2017) de Mikel L. Forcada.

Al igual que los objetivos, la metodología que llevé a cabo se dividió en cuatro fases:

I. Antes de empezar toda la parte práctica del trabajo, realicé algunas lecturas de artículos que introdujeran a un usuario principiante como yo en las tecnologías de la traducción, desde artículos sobre la traducción automática neuronal de mi propio tutor y otros autores hasta trabajos de fin de grado sobre la traducción automática estadística de compañeras de otros años. Las lecturas fueron:
    a. Casacuberta, F. & Peris, A. (2017: 66–74);
    b. Forcada, M. L. (2010: 215–223);
    c. Forcada, M. L. (2017: 291–309);
    d. Forcada, M. L. (2019), manuscrito en revisión;
    e. Forcada, M. L.; Pérez, L. & Rico, C. (2018);
    f. Pérez García, I. M. (2016);
    g. Y Specia, L. & Kashif, S. (2014: 201–235).

    Como se ha comentado en las razones por las que se eligió esta temática, todos estos artículos, excepto el primero, estaban escritos en inglés. Por lo que a estas lecturas les siguió la documentación en distintas páginas de Internet para conocer las definiciones de ciertos conceptos que no me quedaban claros o procesos de la TAN, como la *descodificación* 🕮.
II. Instalación de programas requeridos para la TAN recomendados por mi tutor a la vez que aprendizaje de la utilización del *intérprete de Windows* 🕮 el cual nunca había usado anteriormente. Mientras que yo estaba acostumbrada a instalar programas de la única manera que creía posible, es decir, descargando de una página web el fichero de instalación y siguiendo los pasos, la mayoría de los que usé en este trabajo se instalaron a través del intérprete de mi máquina con la ayuda de órdenes que se encontraban

en guías de Internet o con la ayuda de mi tutor. Los programas requeridos fueron:

a. *Python* 🕮 –el cual incluía *pip* 🕮, otro programa necesario–;
b. La actualización Microsoft Visual C++ 2015 Redistributable Update 3 requerida para la instalación del siguiente programa;
c. El gestor de entornos virtual *Virtualenv* 🕮;
d. La instalación de Git, que permite descargar programas de los repositorios en los que se desarrollan y, además, incluye el intérprete Git Bash y que se necesitaba para la descarga de un archivo `.sh` para la instalación del sistema de TAN vietnamita–inglés y que el intérprete de Windows no era capaz de interpretar;
e. La librería *Tensorflow* 🕮 para la estructura del sistema;
f. El programa de sacremoses para la *tokenización* 🕮 y el proceso de *truecasing* 🕮, y, por último, la instalación de subword-nmt para entrenar con *BPE* 🕮. También se consultaron varias guías en inglés disponibles en la red que explicaban cómo entrenar un sistema de TAN ya creado. Lancé entrenamientos de este sistema instalado a partir de una guía para entrenar un sistema de traducción automática neuronal vietnamita–inglés compuesto por un corpus de 133 000 pares de frases de las *Ted Talks* y con el que pude estudiar el funcionamiento de un sistema de TAN.[10] Aunque no comprendiera el vietnamita, el intérprete mostraba cada cierto número de *pasos* 🕮 una frase del corpus de entrenamiento en vietnamita, otra frase con la traducción de referencia en inglés y otra con la traducción producida por el sistema de TAN y así se podían comparar estas dos últimas para tener una idea de cómo estaba trabajando el

[10] https://github.com/tensorflow/nmt/#hands-on--lets-train-an-nmt-model

sistema. El entrenamiento duraba unas 8 horas. En un entrenamiento tan corto como el de la orden del sistema vietnamita–inglés, tan solo de 12 000 pasos, no se conseguían resultados muy satisfactorios. Los parámetros de la orden[11] que lanzaba el entrenamiento definían la lengua origen, la lengua meta, dónde se encontraban los ficheros de vocabulario, entrenamiento, desarrollo y test, y cómo se llamaban, dónde situar los resultados, el número de iteraciones[12] del entrenamiento, el número de iteraciones que debían pasar para dar estadísticas, el número de capas de la *red neuronal* 🕮, el número de neuronas en cada capa, el porcentaje de neuronas que se eliminan aleatoriamente para hacer más robusto el aprendizaje y el indicador de calidad que se usaba para monitorizar el aprendizaje (*BLEU* 🕮). Muchas veces, particularmente al principio del entrenamiento, el sistema se bloqueaba repitiendo la misma palabra hasta 10 veces seguidas, o adjudicaba etiquetas <*unk*> (del inglés *unknown*) para muchas palabras desconocidas porque no conseguía traducirlas. Después de probar con el vietnamita, decidí pasar a crear un sistema de traducción automática propio que se ajustara a las lenguas de mi interés (francés y español) y utilizar corpus más grandes para poder aplicar el sistema a una situación real.

III. Búsqueda de un posible corpus francés–español para la creación de un sistema de traducción automática neuronal y la posterior creación del sistema. El corpus se descargó de OPUS,[13] una plataforma con numerosos corpus de texto paralelo en gran variedad de lenguas y que alinea las frases de una

---

[11] Diríjase al paso 2 del apartado II. Entrenamiento del sistema de traducción automática neuronal del Apéndice I – Guía de instalación, entrenamiento y uso de su propio sistema de traducción automática neuronal para estudiantes de Traducción e interpretación para ver la orden y sus parámetros.

[12] Sinónimo de pasos.

[13] http://opus.nlpl.eu/

lengua origen con las frases equivalentes de una lengua meta; en mi caso se descargó concretamente de la sección correspondiente a Global-Voices,[14] un portal de noticias redactado por traductores y *blogueros* que dan voz a las personas que no son escuchadas normalmente en los medios de comunicación internacionales, de aquí su nombre. Cuenta con noticias redactadas en 50 lenguas diferentes. Antes de descargarlo había que crear un directorio principal y subdirectorios que albergaran todos los datos y el programa que realiza el proceso de extracción de los corpus de entrenamiento, desarrollo y test a partir del descargado de OPUS, además del preproceso necesario para entrenar sistemas de traducción automática.[15] El programa estaba redactado para generar ficheros de entrenamiento usando palabras completas (sin *BPE* 🕮) o fragmentos de palabras (con BPE) con los que se obtendrían diferentes resultados. Ambas órdenes seguían la misma estructura que la del sistema de TAN vietnamita–inglés con alguna salvedad: el par de lenguas, la localización de los archivos y del destino de los resultados, el número de iteraciones totales (el cual era mayor), y se añadía un parámetro que hiciera que el descodificador «prestara» atención a determinadas posiciones en las representaciones correspondientes a la frase original usando un *modelo determinado de atención* 🕮. Como el entrenamiento con BPE usa subword-nmt,[16] los pasos para entrenarlos son los mismos salvo la modificación de un parámetro de la orden de entrenamiento que lo indica. Además se podían añadir las técnicas de *tokenización* 🕮 (división inteligente de palabras) y *truecasing* 🕮 (normalización de mayúsculas) para mejorar los resultados. Se llamaba al programa Python que leía la orden de entrenamiento y comenzaba con este. Tanto el

[14] http://opus.nlpl.eu/GlobalVoices.php
[15] Creado por Mikel L. Forcada y situado en un proyecto público Gitlab (https://gitlab.com/mlforcada/corpus-utils).
[16] Programa encargado de la segmentación de palabras en unidades más pequeñas (BPE).

entrenamiento sin BPE como con BPE duraron una semana y el *BLEU* 🕮 demostraba que el de BPE tenía mejor calidad. En la siguiente tabla se muestra un ejemplo de traducciones del sistema entrenado sin BPE y con BPE tomadas del conjunto del test:

| SEGMENTO ORIGINAL | SIN BPE | CON BPE |
|---|---|---|
| **GV : À vos yeux quels sont les obstacles à une plus grande utilisation des nouvelles technologies de l'information et de communication (NTIC) dans les processus électoraux en Guinée et dans la sous-région ?** | GV : a tus ojos los obstáculos a una mayor utilización de la tecnología de la información y comunicación ( <unk> ) en los procesos electorales en Guinea y en la <unk> ? | GV : ¿ En qué los obstáculos a una mayor utilización de las nuevas tecnologías de la información y comunicación ( TIC ) en los procesos electorales en Guinea y la subsiña ? |

Se observa claramente la necesidad de llevar a cabo una postedición tanto en la oración meta sin BPE como en la oración meta con BPE, ya sea porque el sistema ha incluido palabras <unk> para las que no encontraba traducción sin BPE (NTIC y *sous-région*), porque se ha inventado una palabra (*subsiña*), construyéndola a partir de trozos de palabras, para una que desconocía (*sous-région*), etc. La ortotipografía no se ha respetado porque eso es tarea del *destokenizador*, es decir, eliminar el espacio entre las palabras y los signos de interrogación o entre las palabras y los dos puntos y seguido.

Una vez que los entrenamientos finalizaron se podían insertar textos en lengua francesa en formato de texto llano (.txt) para que el sistema los tra-

dujera a la lengua española. Se hizo una prueba con un texto de la versión francesa del 26 de mayo de 2019 del portal periodístico GlobalVoices para el cual el sistema estaba expresamente entrenado. Los resultados de dicha prueba se encuentran comentados en el apartado **4. Resultados**.

IV. En paralelo a la fase anterior, iba creando un documento con los problemas, errores, soluciones y pasos que seguía para que funcionaran correctamente cada programa y cada orden. Además, toda esta información me sirvió más tarde para escribir la guía que se encuentra en este trabajo de fin de grado como primer apéndice, el glosario como segundo apéndice y la lista de órdenes como tercer y último apéndice. Al final la guía terminó siendo el eje principal del trabajo.

## 3. Retos

A lo largo del proyecto se hizo frente a tres tipos de retos: aquellos que eran simplemente materiales, los relacionados con el tiempo que tenía para hacer los experimentos, aquellos que estaban en relación con mis conocimientos y habilidades sobre las tecnologías de la traducción y en último lugar los retos o problemas que surgían a lo largo del proceso ya fuera por errores inesperados en el funcionamiento de los programas como por errores de códigos.

En cuanto a los primeros, no poseía un ordenador con mucha memoria, 8 GB de RAM, por lo que iban a surgir limitaciones materiales de hardware. Un factor decisivo para el trabajo es que no tenía una GPU (unidad de procesamiento básico para liberar de trabajo la carga del procesador central) con la que pudiera usar el programa CUDA (componente de software que permite que Tensorflow y otros programas usen la GPU), que ayuda a la máquina a agilizar los entrenamientos, los cuales tardaban una semana en realizarse en la mía, un ordenador SONY Vaio solo con CPU de 64 bits de procesador y 6 núcleos, y procesar corpus mucho más ex-

tensos, de millones de pares de frases, que los que se usaron para esta guía, de 300 000 pares de frases.

Los temporales estuvieron presentes desde el momento en que mi tutor me propuso ir más allá del entrenamiento de un sistema de TAN ajeno y crear un sistema propio para el experimento. En ese momento, fue un aspecto motivador porque para llegar a junio habiendo alcanzado el objetivo, había que trabajar mucho y contrarreloj. Al tener en cuenta las limitaciones materiales que suponía trabajar en un ordenador sin GPU, los entrenamientos aumentaban su duración y el sistema de TAN francés–español necesitaba una semana para cada uno de los entrenamientos, una para BPE y otra para sin BPE. A pesar de que surgieron problemas por el camino tales como errores inesperados de los programas o hacer varias pruebas hasta lanzar los entrenamientos satisfactoriamente, el objetivo se alcanzó satisfactoriamente con la ayuda de mi tutor.

Los retos relacionados con mis conocimientos y habilidades de las tecnologías de la traducción existieron desde el principio, aunque tuviera nociones de inglés, las lecturas se volvían complicadas por toda la terminología inglesa que nunca había escuchado en español y tampoco encontraba gran cantidad de documentos en mi lengua materna para ayudarme a sobrepasar la barrera del idioma. Igualmente se trataba de términos abstractos en su mayoría, por lo que me costó asimilar los conceptos. Terminé asimilándolos a medida que volvía a leer las lecturas iniciales, escuchaba las explicaciones de mi tutor sobre las dudas que tenía, buscaba en Internet intentando quitar al máximo posible el ruido que ofrece el buscador de Google explicaciones básicas de los conceptos o imágenes de los procesos de la traducción automática neuronal. Lo que al principio parecía complicado de asimilar lo fui aceptando gracias a los retos que iban surgiendo en cada sesión de entrenamiento, de instalación o simplemente de reunión con mi tutor y a la constancia de trabajar sobre los mismos conceptos. Una actividad que me ayudó en gran medida fue el hecho de utilizar el intérprete de Windows en las tareas que debía llevar a

cabo. Como tenía que introducir las órdenes yo misma, me esforzaba en entender su construcción y experimentar con ellas para no tener que aprenderlas de memoria y olvidarlas al poco tiempo por no usarlas.

En último lugar, los retos que más me llamaban la atención fueron los de solucionar los problemas y errores que surgían de los programas que utilizaba para los entrenamientos del sistema de TAN. Al principio recurría a mi tutor para conocer el motivo y la solución de los problemas en cuestión, pero conforme avanzaba en el trabajo, me sentía más confiada y segura de que podía resolverlos yo a través de la revisión de mi guía en caso de que ahí estuviera la solución o a través de las publicaciones que realizaban otros usuarios con sus problemas en Internet (normalmente en una sección de la página de *Tensorflow* 🕮) y que si había suerte alguien ya había pasado por donde yo y otra persona le había ayudado a resolverlo. La guía resultante se siguió desde cero en un nuevo ordenador, un portátil Lenovo de 64 bits, para encontrar posibles problemas y que no estuviera personalizada para mi ordenador. Cada paso funcionó correctamente salvo un error que surgió también en mi ordenador personal con el estándar de codificación de caracteres Unicode. El mensaje de error explicaba que no se podía codificar un carácter en especial: el carácter '\u011f' («ğ»). En ambos casos se resolvió de la misma manera: actualizando el programa sacremoses tras haber contactado mi tutor con el autor para explicarle el problema.[17] En el momento de su instalación (al principio de la guía) no se descargó la última versión por lo que había que actualizarla para que no hubiera problema con los códigos. Sin embargo, no fue el único problema que se tuvo con sacremoses. De nuevo, mi tutor contactó con el autor del programa.[18] A la hora de la *destokenización* de una oración por el uso de los apóstrofes en la lengua francesa, el sistema no conseguía mostrar la frase *destokenizada*, por ejemplo, la oración «L'amitié nous a fait forts d'esprit» debería resultar « L' amitié nous

[17] https://github.com/alvations/sacremoses/issues/43
[18] https://github.com/alvations/sacremoses/issues/28

a fait forts d' esprit» (con un espacio tras cada apóstrofe) al ser *destokenizada* pero se obtenía « L & a p o s ; a m i t i é n o u s a f a i t f o r t s d & a p o s ; e s p r i t». La actualización de la versión antigua de sacremoses a una versión más nueva del programa resolvió el problema. Por último, el *intérprete de órdenes* 🕮 sacó un error cuando se probó Tensorflow[19] antes de realizar los entrenamientos sin BPE y con BPE en el que explicaba que no encontraba un paquete en concreto (*numpy*) que el sistema necesitaba importar para lanzar el entrenamiento. Tardé en dar con la solución porque no era lógica: aunque Tensorflow estuviera actualizado, ese mismo paquete que no encontraba el sistema, formaba parte a su vez de las dependencias del paquete de Tensorflow, pero no se había actualizado automáticamente a la vez que este último. Por lo que a través de una orden de actualización, se instaló una versión más nueva de ese paquete y se solucionó el problema.

A mitad del trabajó se pensó en intentar utilizar una versión de Tensorflow para GPU aprovechando la capacidad de 4 GB de mi GPU. Para ello se tuvo que instalar CUDA, teniendo en cuenta el tipo de tarjeta gráfica de mi ordenador (GeForce 740M) y sus librerías CuDNN, se eligieron las versiones 10 para CUDA y 7.3.0 para CuDNN pero en el momento de importar Tensorflow para comprobar que se había instalado correctamente, surgió un error de versiones. Este error se solventó con el cambio a una versión más antigua de CUDA (la 9 exactamente) y a su vez de CuDNN (la 7.3.1). El entrenamiento con la prueba de la GPU se lanzó con el sistema vietnamita–inglés, y aunque se fueron disminuyendo los parámetros del tamaño del lote (*batch size*)[20] hasta llegar al mínimo, el entrenamiento nunca se pudo llegar a empezar debido a un problema de falta de memoria. Como mi GPU

---

[19] Comprobación a través de una orden para confirmar que no existe ningún problema con el programa. Ver Apéndice I – Guía de instalación, entrenamiento y uso de su propio sistema de traducción automática neuronal para estudiantes de Traducción e interpretación.

[20] Término utilizado en el aprendizaje automático que se refiere al número de ejemplos de entrenamiento utilizados antes del cálculo de cada paso o iteración, o sea, de una actualización de los pesos.

no tenía suficiente potencia para usar la versión de Tensorflow para GPU, el experimento tuvo que dejarse a un lado.

## 4. Resultados

En este apartado se comentarán los resultados obtenidos por el sistema de TAN francés–español de textos periodísticos.

En el momento en que el sistema acabó con sus entrenamientos, sin BPE y con BPE, este estaba listo para llevar a cabo una tarea realista: introducir un nuevo texto en lengua francesa desconocido para él y en formato de texto llano (.txt) y que a partir de este produjera su traducción en bruto al español. Puesto que la plataforma a la que pertenecían era de un portal de noticias, GlobalVoices, se hizo una prueba con un texto de ese mismo portal para el cual el sistema había sido expresamente entrenado, de un extracto de un artículo de la versión francesa del 26 de mayo de 2019. El texto estaba compuesto por las siguientes tres oraciones:

> Youtubeurs, journalistes et politiciens unissent leurs forces contre les assassinats extrajudiciaires de leaders sociaux (ou défenseurs sociaux, selon un autre terme, NdT) en Colombie. Avec la campagne #UnChefAMaPlace, ils ouvrent leurs espaces médiatiques pour permettre aux leaders et défenseurs de droits menacés d'écrire ou d'exprimer leurs opinions. La campagne a démarré le 13 mai, lorsque le journaliste Daniel Samper Ospina a posté sur sa chaîne YouTube un clip d'une chanson de reggaeton avec des dirigeants sociaux et de célèbres YouTubeurs.

Antes de traducirlo se preparaba el texto con la *tokenización* 🕮 y *truecasing* 🕮 obteniendo un texto con los signos de puntuación y los apóstrofes separados de las palabras y con las palabras en sus mayúsculas y minúsculas naturales obtenidas por el programa sacremoses. Además, se dividieron las palabras en unidades BPE con el programa subword-nmt en unidades que son susceptibles de repetirse más a menudo que las palabras completas, y así mitigar el problema que comporta traducir palabras que no se hayan visto durante el entrenamiento. La traducción en bru-

to tenía que ser pasar por un *detruecasing* y una *destokenización* para juntar las palabras con los signos de puntuación y recolocar las mayúsculas. El texto meta producido por el sistema entrenado sin BPE presentaba algunas palabras desconocidas (casi todas eran nombres propios):

> <unk>, periodistas y políticos <unk> sus fuerzas contra los asesinatos <unk> de líderes sociales (o defensores sociales), según otro término, <unk>) en Colombia. Con la campaña # <unk>, <unk> sus espacios de medios para permitir a los líderes y defensores de derechos humanos que <unk> o expresar sus opiniones. La campaña comenzó el 13 de mayo cuando el periodista Daniel <unk> <unk> publicó en su canal YouTube un video de una canción de <unk> con líderes sociales y famosos <unk>.

Mientras que el entrenamiento con BPE encontraba traducciones para las que el anterior no tenía solución aunque se inventara palabras (*extrajudiciarios*). Aunque sí analice las palabras que están incluidas en el *hashtag* (#), no las traduce por la campaña colombiana #UnLíderEnMiLugar y las deja en la LO. A simple vista pueden verse pequeños errores que necesitan ser posteditados: el uso incorrecto de los tiempos verbales (el subjuntivo *abran*), la inclusión de complementos de régimen en un contexto inadecuado (unirse a ≠ unir algo) o la omisión de acentos (*video*, *reggaeton*).

> Los periodistas y políticos unan a sus fuerzas contra los asesinatos extrajudiciarios de líderes sociales (o defensores sociales, según un nuevo término, NdT) en Colombia.
>
> Con la campaña #UnChefAMaPlace, abran sus espacios mediáticos para permitir a los líderes y defensores de derechos de escribir o expresar sus opiniones.
>
> La campaña comenzó el 13 de mayo cuando el periodista Daniel Samper Ospina publicó en su canal de YouTube un video de una canción de reggaeton con líderes sociales y famosos YouTubadores.

## 5. Conclusión

Este trabajo me ha hecho reafirmarme en la idea de que el buen dominio de las tecnologías de la traducción por parte de un traductor es más que necesario y que negarse a los avances que está experimentando la TA es ponerse barreras a su propia carrera profesional. Como se puede ver en el ejemplo expuesto en el apartado anterior, los resultados que puede obtener un sistema de TAN entrenado con corpus de un tamaño mediano son muy prometedores, ¿por lo que cómo llegarían a ser si fueran corpus grandes? Se trata de un experimento que renueva constantemente los objetivos, es decir, no tiene porqué acabar aquí, sino que se puede continuar más allá de este trabajo de fin de grado. Una posible continuación de este trabajo supondría la creación de mi propio corpus jurídico a partir de las traducciones de aspecto jurídico que se han hecho durante los dos últimos cursos del grado y con otros corpus descargados de Internet, y con él entrenar un sistema de TAN para futuros encargos jurídicos. Puesto que el corpus no sería lo suficientemente grande, se podrían obtener de Internet textos de esta temática, por ejemplo, los documentos de la ONU (acuerdos internacionales, directivas, reglamentos, dictámenes, entre otros) que están disponibles en francés y español, entre otros idiomas. Un ejemplo, serían los corpus disponibles en OPUS: EUconst (de la Constitución Europea),[21] EUROPARL (del Parlamento Europeo)[22] y JRC-Acquis (textos legislativos de la Unión Europea).[23] He disfrutado mucho resolviendo los problemas que iban surgiendo o dejándome guiar por mi tutor para conseguir resultados y ahora me llevo una opinión mucho más justa y merecida de esa zona en la que conviven la traducción y la informática.

El objetivo que se pretendía cumplir al haber redactado esta guía y los apéndices que la completan no es otro que acercar la traducción automática neuronal a los alumnos de la carrera de traducción, y también a aquellos que ya acabaron su

---

[21] http://opus.nlpl.eu/EUconst.php
[22] http://opus.nlpl.eu/Europarl.php
[23] http://opus.nlpl.eu/JRC-Acquis.php

formación académica y quieren adentrarse en el mundo de la TA. Es decir, facilitar el proceso de comprensión de su funcionamiento y de los conceptos básicos que la rodean e invitar al lector a probar él mismo cómo hacerlo antes de pensar que no es capaz, pensamiento que lamentablemente comparten muchos compañeros de traducción. Espero haber contribuido con mi proyecto a este objetivo.

## 6. Referencias bibliográficas


Casacuberta Nolla, F. & Peris Abril, A. (2017) "Traducción automática neuronal." Revista *Tradumàtica. Tecnologies de la Traducció 15*, pp. 66-74 Versión electrónica: <https://ddd.uab.cat/pub/tradumatica/tradumatica_a2017n15/tradumatica_a2017n15p66.pdf>.

Crystal, D. (2000) *Diccionario de lingüística y fonética*. Barcelona: Octaedro.

Forcada, M. L. (2010) "Machine translation today." En Yves Gambier, Luc Van Doorslaer (eds) 2010. *Handbook of Translation Studies 1*. John Benjamins Publishing Company, pp. 215–223.

Forcada, M. L. (2017) "Making sense of neural machine translation." *Translation Spaces 6:2*, pp. 291-309. Versión electrónica: <http://hdl.handle.net/10045/87067>.

Forcada, M. L.; Pérez, L. & Rico, C. (2018) *Translators' track: a glossary*. Alicante: EAMT. Versión electrónica: <http://eamt.org/translators_documents/eamt_2018_glossary.pdf>.

Forcada, M. L. (2019) "Què és la traducció automàtica neural i què comporta per a la traducció professional?" Manuscrito en revisión, cedido por el autor.

Hutchins, W. J. & Somers H. L. (1995) *Introducción a la traducción automática*. Madrid: Visor distribuciones.

Mikolov, T.; Wen-tau Y. & Geoffrey Z. (2013b) "Linguistic Regularities in Continuous Space Word Representations." En: NAACL-HLT 2013 (Atlanta, Georgia, 9–14 junio).

Pérez García, I. M. (2016) *Phrase-based statistical machine translation: explanation of its processes and statistical models and evaluation of the English to Spanish translations produced*. Alicante: Universidad de Alicante. Departamento de Lenguajes y Sistemas Informáticos. Versión electrónica: <http://hdl.handle.net/10045/56346>.

Specia, L. & Kashif, S. (2014) “Machine Translation Quality Estimation: Applications and Future Perspectives.” En: Moorkens J., Castilho S., Gaspari F., Doherty S. (eds) 2014. *Translation Quality Assessment. Machine Translation: Technologies and Applications*. Springer: Cham, pp. 201–235.

# APÉNDICES

# Apéndice I – Guía para instalar, entrenar y usar su propio sistema de traducción automática neuronal para estudiantes de Traducción

Clara García Abiétar

02/06/2019

*https://drive.google.com/open?id=1tTPkicaIAreLgF1EF0j84zkVkOL0aWVb*

Esta guía surge de la escasez de información y de apoyo didáctico en el campo de las tecnologías de la traducción en lengua española. A lo largo de la guía se le explicarán los pasos necesarios para utilizar su propio sistema de traducción automática neuronal, y para ello se describirán los siguientes apartados: a) instalación de TensorFlow en un ordenador Windows, b) entrenamiento de un sistema de traducción automática neuronal –más adelante, TAN– y c) aprendizaje para generar sus propias traducciones. En Internet se puede encontrar un gran número de páginas, sobre todo en inglés, que explican aspectos de la TAN o de cómo funcionan, por lo que me he apoyado en algunas de estas como: la web oficial de TensorFlow[1] y en un tutorial de un TAN para TensorFlow de GitHub;[2] a las que podrá recurrir igualmente, en caso de que a lo largo de esta guía le surgiera alguna duda.

La TAN es un tipo de TA basada en corpus que utiliza redes neuronales artificiales entrenadas sobre corpus paralelos de miles o millones de pares de frases. Una gran parte de los sistemas de TAN consisten en un codificador y un descodificador, cada uno de los cuales es una red neuronal (recurrente) artificial especializada. La activación de cada neurona artificial de la red (su excitación o depresión) depende de la activación de las otras neuronas y del peso de las conexiones. Así el signo y la magnitud del peso de la conexión determina el comportamiento de la

---

[1] https://www.tensorflow.org/install/pip

[2] https://github.com/tensorflow/nmt/#hands-on--lets-train-an-nmt-model

red neuronal. Si el peso es positivo, las neuronas tienden a excitarse o deprimirse la una a la otra. Mientras que si el peso es negativo, las neuronas conectadas tienden a permanecer en el estado opuesto. Este efecto aumenta con la magnitud del peso. El entrenamiento consiste en fijar las cargas a los valores necesarios para así asegurar un comportamiento determinado.

Estos experimentos se han hecho en un ordenador portátil Sony Vaio con 6 *cores* (CORE™ i7) y 8 GB de memoria RAM y un sistema operativo de 64 bits. En teoría, un sistema se puede entrenar en cualquier máquina, pero para realizar entrenamientos en un espacio de tiempo razonable, se recomienda un equipo especializado con una o más GPU instaladas –usando a veces hasta 64 GB de memoria– que además pueda soportar CUDA, una plataforma donde los desarrolladores disponen de herramientas para acelerar considerablemente las aplicaciones informáticas aprovechando la potencia de las GPU. Mientras que este proyecto se ha realizado con ordenadores que disponían solamente de CPU.

Si tuviera alguna duda relacionada con los conceptos de la guía, tiene disponible el Apéndice I – Glosario. Todas las órdenes utilizadas en esta guía están explicadas de manera más exhaustiva en el Apéndice II – Órdenes para Bash y para *Shell* de Windows. Ambos archivos pueden descargarse junto con esta guía a través del enlace Drive que se sitúa en la información debajo del título de este documento.

# I. Instalación de los programas necesarios

TensorFlow es una *librería* (un anglicismo de *library*, biblioteca de programas) de código abierto, es decir, se puede acceder a ella libremente y un programador puede modificarla según las necesidades del usuario. Tensorflow permite construir, entrena y usar redes neuronales artificiales. Este sistema es un *encoder–decoder*, una posible arquitectura para los sistemas de TAN.

Los pasos para instalar Tensorflow son los siguientes:

1. En el buscador de Windows, escriba CMD o Símbolo del sistema para acceder al intérprete de Windows y en él compruebe si hay instalada alguna versión de los programas necesarios para crear el entorno de Python:

    a. Escriba `python --version` para saber si Python se encuentra instalado y, en caso afirmativo, qué versión. En algunos tutoriales, también encontrará este comando como `python3 –version`, aunque normalmente se sobreentiende que está instalada la versión 3 de Python, al ser la más reciente.

    b. A continuación, `pip --version` para saber qué versión de pip.

    c. Y por último, `virtualenv --version` para el entorno virtual.

2. Si estos programas están instalados, diríjase al paso 3. Si no están instalados, es necesario hacerlo. Para instalar Python, en uno de los apartados de su página web[3] se encuentran todas las versiones disponibles para Windows; usted optará por la última versión para x86-64 executable installer (la 3.7.2 en el momento de escribir estas líneas), si su PC es de 64 bits, como será probablemente hoy en día. Tras la instalación, es muy importante cerrar el intérprete de órdenes (`exit`) donde se ha instalado Python y volver a abrir uno nuevo; si no, el sistema podría no encontrar la versión de Python y hacerle pensar que ha surgido algún error en la instalación. En la nueva ventana del intérprete de Windows, compruebe con las órdenes de los pasos 1.a y 1.b si se ha instalado correctamente y, debido a que en las últimas versiones de Python pip[4] ya está integrado, nos saldrá como instalado. De todas formas, es posible que sea necesario actualizar pip introduciendo en la línea de órdenes `python -m pip install -U`

[3] https://www.python.org/downloads/windows/
[4] https://pip.pypa.io/en/stable/installing/

`pip`. El último paso es la instalación del gestor de entornos virtuales Virtualenv, una herramienta que ayuda a crear instalaciones separadas cuando las dependencias requeridas por proyectos diferentes son asimismo diferentes. Esta es una de las herramientas más importantes que la mayoría de los desarrolladores de Python utilizan. Para ello necesitamos instalar antes uno de los paquetes que integra Visual Studio. Puede que su ordenador ya lo tenga; si no, en la página de Visual Studio de Microsoft,[5] ofrecen por separado el paquete que se necesita: la versión x64 de «Microsoft Visual C++ 2015 Redistributable Update 3», si se tiene un PC de 64 bits, en el último apartado denominado «Redistribuibles y herramientas de compilación». Cuando ya se ha instalado este paquete, escriba en el símbolo del sistema `pip3 install -U pip virtualenv`. Ya estarán instalados los tres paquetes necesarios para poder proceder a la instalación de TensorFlow.

3. Cree un entorno virtual dentro de su directorio de usuario. Por ejemplo, `virtualenv --system-site-packages -p python ./prueba`, siendo *prueba* el nombre del entorno virtual (puede poner cualquier otro nombre). Acceda a este nuevo directorio (`cd prueba`) y active el entorno virtual correspondiente a través de `.\Scripts\activate`.

4. Para salir de este entorno virtual basta con escribir `deactivate`. Cuando esté usando TensorFlow, no debería salir del entorno hasta que no haya acabado.

5. Ahora ya se puede comenzar con la instalación de TensorFlow. Desde el intérprete de órdenes (CMD) de su ordenador, dentro del entorno virtual

---

[5] https://visualstudio.microsoft.com/es/vs/older-. downloads/?rr=https%3A%2F%2Fwww.tensorflow.org%2Finstall%2Fpip%3Flang%3Dpython3

creado y una vez activado (paso 3), escriba `pip install tensorflow`. Y compruebe con `pip list` que posee la versión más actualizada de TensorFlow (actualmente 1.13.1).

6. Para saber si TensorFlow funciona correctamente, hay dos posibles comprobaciones:

    a. Ejecute un intérprete de Python escribiendo `python` e importe TensorFlow con `import tensorflow as tf`. Si no comunica ningún error, TensorFlow se ha instalado correctamente. Acabe con esta orden para salir del intérprete de Python (`exit()`).

    b. Dentro del entorno virtual activado, ejecute `python -c "import tensorflow as tf; tf.enable_eager_execution(); print(tf.reduce_sum(tf.random_normal([1000, 1000])))"`; si no comunica ningún error, Tensorflow se ha instalado correctamente.[6]

7. En la ventana del intérprete de Windows, acceda al entorno que ha creado anteriormente (`cd prueba`), suponiendo que lo ha denominado *prueba*, e introduzca la orden `git clone https://github.com/tensorflow/nmt/`. No es necesario activar el entorno para este paso.

8. Para generar los ficheros de entrenamiento, desarrollo y test en el siguiente apartado, se necesitan dos programas para que se creen correctamente: *sa-*

[6] Si no se hubiera instalado correctamente después de comprobarlo de las dos maneras distintas, puede ser que deba instalar una versión específica de Tensorflow para su ordenador concreto. Al ser esta una situación poco común, debería ponerse en contacto con una persona experta en informática.

*cremoses*, encargado de la *tokenización* (y *destokenización*) y del proceso de *truecasing* (el del *detruecasing*), y *subword-nmt*, encargado de la segmentación de palabras (BPE). Para esto, abre un nuevo intérprete, ejecuta, activando el entorno virtual correspondiente (`.\Scripts\activate` si está situado en el directorio prueba) ambas instalaciones con `pip install sacremoses` y después `pip install subword-nmt`. Compruebe que tiene una versión igual o mayor a 0.0.19 en *sacremoses* (`sacremoses --version`) para que no surjan problemas con la codificación de caracteres. Para asegurarse de que tiene una versión reciente de *sacremoses* y de que no le surgirá ningún problema con Unicode a la hora de entrenar su sistema, deberá actualizarlo a través de `pip install -U sacremoses`. Una versión anterior a la 0.0.8. dará problemas de codificación.

9. Antes de empezar con el primer entrenamiento de la siguiente sección, es necesario instalar Git de 64 bits para Windows,[7] el cual incluye el intérprete de órdenes Git Bash que permite descargar los corpus necesarios que proporcionan con libre acceso y entrenar un pequeño sistema de traducción automática neuronal puesto que se trata de la descarga de un archivo `.sh` y el *shell* de Windows no reconoce este tipo de archivos. Si por el contrario, pasa directamente a la sección del sistema de traducción francés–español, esta instalación no será necesaria. La instalación de Git incluye el intérprete de orden Git Bash. Bash es uno de los intérpretes de órdenes de Unix/Linux más populares, y Git Bash es un intérprete basado en bash compatible con Windows. En esta instalación, tiene que elegir entre varios parámetros, dejará las opciones predeterminadas: «Git from the command line and also from 3rd-party software», «Use the OpenSSL li-

[7] https://git-scm.com/download/win

brary», «Checkout Windows-style, commit Unix-style line endings» y, por último, «Use MinTTY (the default terminal of MSYS2)».

> NOTA
> Es importante saber que los intérpretes de órdenes o *shells* de los distintos sistemas operativos tienen distintas órdenes. Por ejemplo, GNU/Linux usa la barra inclinada / para las rutas de los archivos y directorios, entre otros. Sin embargo, Windows usa la barra invertida \. Un ejemplo en Windows sería `.\Scripts\activate`, pero, en Git Bash de Windows sería `source ./Scripts/activate` y en el *shell* de GNU/Linux: `source ./bin/activate`.

## II. Entrenamiento del sistema de traducción automática neuronal

En este primer apartado, se explica y desarrolla el proceso de entrenamiento de un sistema de traducción automática neuronal a partir de un corpus vietnamita–inglés de las *TED Talks*, el mismo corpus que se utilizó al principio para aprender a crear un sistema de traducción automática neuronal. Su entrenamiento está desarrollado paso por paso al igual que se hizo para redactar esta guía en caso de que usted se animase a probarlo.

1. El siguiente paso es acceder a la carpeta nmt del repositorio de prueba (`cd nmt`) y ejecutar nmt/scripts/download_iwslt15.sh /tmp/nmt_data, puesto que se trata de un archivo `.sh` (download_iwslt15.sh).[8] Sin embargo, el intérprete de órdenes de Windows no es capaz de ejecutar el guion de ór-

[8] Es decir, se trata de un guion de órdenes de intérprete de órdenes que puede ser ejecutado por Bash, entre otros intérpretes.

denes `iwslt15.sh` y este paso debe realizarse con el intérprete alternativo Git Bash. Este archivo nos descarga un corpus vietnamita–inglés de las TED Talks para entrenar el sistema de TAN. Este corpus ya contiene los corpus de desarrollo, entrenamiento y test puesto que se crean con la orden anterior (`download_iwslt5.sh`) que sin embargo hay que crear en el momento en que se quiere crear y entrenar un sistema propio como se desarrolla más adelante. Puede encontrar un tutorial en lengua inglesa en el apartado Hands on - Let's train an NMT model del tutorial de GitHub.[9] Una vez instalado, salimos de Git Bash (exit).

2. Ya puede empezar a entrenar su sistema de TAN. Para ello, debe volver al intérprete de órdenes de Windows, acceder a su entorno, activarlo y acceder a la carpeta creada con Git Bash llamada nmt (tres órdenes `cd prueba`; `.\Scripts\activate`; `cd nmt`) e incluir la siguiente orden directamente en el intérprete para crear un fichero temporal:

   ```
   mkdir \tmp\nmt_model
   ```

   Seguida de la siguiente:

   ```
   python -m nmt.nmt ^
       --src=vi --tgt=en ^
       --vocab_prefix=/tmp/nmt_data/vocab ^
       --train_prefix=/tmp/nmt_data/train ^
       --dev_prefix=/tmp/nmt_data/tst2012 ^
       --test_prefix=/tmp/nmt_data/tst2013 ^
       --out_dir=/tmp/nmt_model ^
       --num_train_steps=12000 ^
       --steps_per_stats=100 ^
       --num_layers=2 ^
   ```

[9] https://github.com/tensorflow/nmt/#hands-on--lets-train-an-nmt-model

```
--num_units=128 ^
--dropout=0.2 ^
--metrics=bleu
```

Esta orden también puede copiarse en un fichero de guion de órdenes (ordenesvien.bat). El entrenamiento en el ordenador utilizado para el desarrollo de este proyecto dura unas 8 horas. La orden indica al intérprete que la lengua origen es vietnamita (src=vi) que la lengua meta es inglés (tgt=en), dónde se encuentran los ficheros y cómo se llaman (quitándoles las terminaciones de las lenguas, «.vi» y «.en»; el del vocabulario vocab_prefix=/tmp/nmt_data_vocab, el del entrenamiento train_prefix=/tmp/nmt_data/train, el del desarrollo dev_prefix=/tmp/nmt_data/tst2012 y el del test test_prefix=/tmp/nmt_data/tst2013), en qué directorio situar los resultados (out_dir=/tmp/nmt_model), el número de iteraciones totales que realizará el entrenamiento (num_train_steps=12 000), el número de iteraciones que deben pasar para dar estadísticas (steps_per_stats=100) el número de capas de la red (num_layers=2, tanto el codificador como el descodificador), el número de neuronas en cada capa (num_units=128), el porcentaje de neuronas que se eliminan aleatoriamente y periódicamente para hacer robusto el aprendizaje (dropout=0.2) y, por último, el indicador de calidad que se usará para monitorizar el aprendizaje (metrics=bleu).

Con esta orden el intérprete refleja en la pantalla los resultados del entrenamiento del sistema, cada X pasos muestra el segmento original en lengua vietnamita, el segmento en la lengua inglesa ya traducido procedente del corpus y el segmento en lengua inglesa propuesto por el sistema de TAN. Los resultados son interesantes porque se pueden comparar los dos últimos segmentos si el usuario tiene menciones de inglés, pero a la hora de compararlo con el segmento original para conocer más a fondo el senti-

do de la oración, si no se tiene nociones de vietnamita, no se puede estar completamente seguro de la calidad de la traducción.

> NOTA
> En la página mencionada más arriba del tutorial de GitHub, encontrará esta orden con las barras invertidas al final de cada línea, para usarla con el intérprete Bash. En Windows, para que el intérprete de órdenes entienda el salto de línea, es necesario hacerlo con el acento circunflejo (`^`).

En este segundo apartado, le explicaré cómo entrenar el sistema de traducción automática neuronal que creé a partir de un corpus francés–español de textos periodísticos de la plataforma GlobalVoices[10].

1. Crear a través del CMD con `md sistema` un archivo denominado *sistema* para albergar los archivos de nuestro sistema de TAN en su directorio principal de usuario (C:\Users\clara en este proyecto).
2. Entre en este directorio y cree los siguientes:
    a. RAWDATA con md RAWDATA;
    b. DATA con md DATA;
    c. DATABPE con md DATABPE;
    d. MODEL con md MODEL;
    e. MODELBPE con md MODELBPE.
3. Tendrá que descargarse los corpus de OPUS,[11] una plataforma con numerosos corpus de texto en gran variedad de lenguas y que alinea las frases

[10] https://globalvoices.org/

de una lengua origen con las frases equivalentes de una lengua meta. A través del enlace[12] se descargará los textos paralelos de español–francés de la plataforma periodística GlobalVoices en el denominado formato Moses,[13] es decir, un total de dos ficheros, uno en español y otro en francés, con una oración por línea. Será un archivo comprimido, por lo que deberá descomprimirlo seleccionando como carpeta de destino RAWDATA creada en el anterior paso para albergar los archivos descomprimidos. En este directorio habrá, entre otros, dos ficheros, uno para cada idioma, GlobalVoices.es-fr.es y GlobalVoices.es-fr.fr, los cuales tienen 321 017 pares de segmentos, de los que se elegirán aleatoriamente 250 000 para el entrenamiento, 1 500 para el desarrollo y 1 500 para el test.

4. Dentro del directorio sistema creado en el paso 1, descargue el archivo prepare.py haciendo `git clone https://gitlab.com/mlforcada/corpus-utils`.[14] Este archivo le ayudará a crear los tres corpus (entrenamiento, desarrollo y test); este programa necesita los programas *sacremoses* y *subword-nmt*, instalados en el paso 9 de la fase I. Instalación de los programas necesarios. Verá que se ha descargado un directorio con el nombre *corpus-utils*, cámbielo a CODE (`rename corpus-utils CODE`).

A partir de este momento, se pueden lanzar dos tipos de entrenamientos: un entrenamiento sin BPE y otro con BPE.[15] Se obtendrán diferentes resultados que

---

[11] http://opus.nlpl.eu/
[12] http://opus.nlpl.eu/download.php?f=GlobalVoices/v2017q3/moses/es-fr.txt.zip
[13] Llamado así por ser el formato que se usaba para entrenar este sistema de traducción automática estadística.
[14] Este programa ha sido creado por Mikel Forcada y realiza el proceso de extracción de los corpus de entrenamiento, desarrollo y test, además del preproceso necesario para entrenar sistemas de traducción automática.
[15] Diríjase al Apéndice I – Glosario para conocer el significado de BPE.

compararé más adelante y que si realiza ambos entrenamientos podrá ver usted mismo.

A) ENTRENAMIENTO SIN BPE

1. Entre a DATA (`cd DATA`) ejecute la siguiente orden para crear los ficheros *tokenizados* y con las palabras en mayúsculas naturales (formato *truecase*): `python ../CODE/corpus-utils/prepare.py ../RAWDATA/GlobalVoices.es-fr fr es 260000 1500 1500 10000 --tokenize --truecase`. Estos datos corresponden a: el prefijo común a los nombres de fichero del corpus (GlobalVoices.es-fr), la lengua 1 (fr), la lengua 2 (es), el tamaño del corpus de entrenamiento (260 000), el tamaño del corpus de desarrollo (1 500), el tamaño del corpus de test (1 500), el tamaño del vocabulario (10 000), el *tokenizador*, el *truecase* y, por último, el BPE. El *tokenizador* se encarga de segmentar el texto original en *tokens*, generalmente en palabras o unidades similares, y que tiene que deshacer antes de obtener el texto meta; el *truecaser* de colocar las palabras en su forma más probable (cambiará las mayúsculas a minúsculas, cuando sea necesario) y el BPE de descomponer las palabras en unidades que son susceptibles de repetirse más a menudo que las palabras completas. La importancia de estos tres corpus se debe a que el traductor automático necesita oraciones paralelas en lengua origen y en lengua meta para entrenarse. El corpus de entrenamiento es el más grande (entre 100 000 y 1 000 000 de pares de frases) con el que se entrena, mientras que el de desarrollo con el que se determina cuándo dejar de entrenar para no «sobreentrenarlo» (memorización de los ejemplos e impide al sistema generalizar lo aprendido a otras frases), y el de test deben tener entre 1 000 y 3 000 pares para hacerse una idea del rendimiento del sistema.

2. Una vez terminado el paso anterior, ya se puede empezar a entrenar el sistema. Salga de DATA (`cd ..`) y de sistema (`cd ..`) y acceda a la carpeta prueba (`cd prueba`) y active el entorno correspondiente (`.\Scripts\activate`). Después, entre en la carpeta nmt (`cd nmt`) que se creó en el paso 8 de la fase I. Instalación de los programas necesarios con el git clone y lance el siguiente entrenamiento:[16]

```
python -m nmt.nmt ^
  --attention=scaled_luong ^
  --src=fr ^
  --tgt=es ^
  --vocab_prefix=C:\Users\clara\sistema\DATA\vocab.tok.true ^
  --train_prefix=C:\Users\clara\sistema\DATA\train.tok.true ^
  --dev_prefix=C:\Users\clara\sistema\DATA\dev.tok.true ^
  --test_prefix=C:\Users\clara\sistema\DATA\test.tok.true ^
  --out_dir=C:\Users\clara\sistema\MODEL ^
  --num_train_steps=250000 ^
  --steps_per_stats=1000 ^
  --num_layers=2 ^
  --num_units=128 ^
  --dropout=0.2 ^
  --metrics=bleu
```

Esta orden también puede copiarse en un fichero de guion de órdenes (por ejemplo, *ordenessin.bat*). Con la capacidad del ordenador utilizado en este proyecto, el entrenamiento duraba una semana. La orden es similar a la orden de entrenamiento de un sistema de traducción vietnamita–inglés con algunas salvedades: el par de lenguas (esta vez francés–español), la localización de los archivos es distinta al igual que el

[16] Debe tener en cuenta que en los parámetros de esta orden se encuentran las localizaciones de mi sistema (desde el parámetro de vocab_prefix hasta el out_dir, por ejemplo, `C:\Users\clara\sistema\DATA\vocab.tok.true`) y puede que en su máquina no sea la misma, verifíquelas antes de lanzar el entrenamiento.

fichero destino de los resultados, el número de iteraciones totales que realizará el entrenamiento es mayor, y el parámetro que se le indica al intérprete para que el descodificador «preste» atención a determinadas posiciones en las representaciones correspondientes a la frase original usando un modelo determinado de atención (attention=scaled_luong).[17]

B) ENTRENAMIENTO CON BPE

1. Si ha hecho el entrenamiento sin BPE, salga de cd DATA (`cd ..`), entre a DATABPE (`cd DATABPE`) y ejecute la siguiente orden para crear los ficheros *tokenizados* y con las palabras en mayúsculas naturales (formato *truecase*): `python ../CODE/corpus-utils/prepare.py ../RAWDATA/GlobalVoices.es-fr fr es 260000 1500 1500 10000 --tokenize --truecase --bpe`. Si ha empezado directamente por este entrenamiento, salga del directorio CODE (`cd ..`), entre a DATA (`cd DATA`) y ejecute la orden anterior.
2. Una vez terminado el paso anterior, ya se puede empezar a entrenar el sistema. Salga de DATA (`cd ..`) y de sistema (`cd ..`) y aceda a la carpeta prueba (`cd prueba`) y active el entorno correspondiente (`.\Scripts\activate`). Después, entre en la carpeta de nmt (`cd nmt`) que se creó en el paso 8 de la fase I. Instalación de los programas necesarios con el git clone y lance el siguiente entrenamiento:[18]

```
python -m nmt.nmt ^
    --attention=scaled_luong ^
```

[17] Si no, prestaría atención únicamente a la representación obtenida al procesar toda la oración original.

[18] Debe tener en cuenta que en los parámetros de esta orden se encuentran las localizaciones de mi sistema (desde el parámetro de vocab_prefix hasta el out_dir, por ejemplo, `C:\Users\clara\sistema\DATABPE\vocab.tok.true`) y puede que en su máquina no sea la misma, verifíquelas antes de lanzar el entrenamiento.

```
--src=fr ^
--tgt=es ^
--subword_option=bpe ^
--vocab_prefix=C:\Users\clara\sistema\DATABPE\vocab.bpe ^
--
train_prefix=C:\Users\clara\sistema\DATABPE\train.tok.tru
e.bpe ^
--
dev_prefix=C:\Users\clara\sistema\DATABPE\dev.tok.true.bp
e ^
--
test_prefix=C:\Users\clara\sistema\DATABPE\test.tok.true.
bpe ^
--out_dir=C:\Users\clara\sistema\MODELBPE ^
--num_train_steps=250000 ^
--steps_per_stats=1000 ^
--num_layers=2 ^
--num_units=128 ^
--dropout=0.2 ^
--metrics=bleu
```

Al igual que la orden para el entrenamiento sin BPE, esta orden también puede copiarse en un fichero de guion de órdenes (por ejemplo, *ordenesbpe.bat*). Este entrenamiento duraba también aproximadamente una semana. La orden contiene los mismos parámetros que la orden sin BPE, pero con localizaciones distintas de los archivos requeridos y del directorio de destino (C:\Users\clara\sistema\DATABPE y C:\Users\clara\sistema\MODELBPE).

Más adelante se presenta una comparación de los resultados obtenidos. Las traducciones del fichero test.fr aparecen en los directorios MODEL y MODELBPE en el archivo output_test. Para conocer los valores de BLEU[19] alcanzados con los

[19] Diríjase al Apéndice I – Glosario para conocer la definición de BLEU.

parámetros de ambos entrenamientos, debe dirigirse al conjunto de test puesto que es el conjunto que nos proporciona una idea independiente del rendimiento del sistema ya que no se usa para entrenarlo o para guiar su entrenamiento:

1. Sin BPE los mejores valores obtenidos de BLEU se posicionan en el paso 240 000 de un total de 250 000 con 21.3 en el corpus de desarrollo y 20.9 en el del test.
2. Con BPE los mejores valores obtenidos de BLEU se posicionan en el paso 127 000 de un total de 250 000 con 22.5 en el corpus de desarrollo y 22.1 en el del test.

Esta diferencia de 2.2 puntos en BLEU entre el resultado de desarrollo sin BPE y con BPE podría ser significativa en algunas aplicaciones. Además, la salida es bastante diferente. Cuando no se usa BPE, aparecen traducciones con palabras desconocidas que el sistema de traducción etiqueta como <unk> (del inglés *unknown*). Sin embargo con BPE el sistema construye palabras incorrectas a partir de unidades sub–palabra (como se explicará más abajo). A continuación, se muestra la comparación de ejemplos de traducciones sin BPE y con este, tomadas del conjunto de test.

Ejemplo 1:

| SEGMENTO ORIGINAL | SIN BPE | CON BPE |
|---|---|---|
| **Elle estime qu'environ 40 pour cent ou plus des plastiques agricoles collectés dans le but d'être recyclés sont exportés, communément en Chine** | ella estima que alrededor del 40 % o más de los plásticos agrícolas <unk> en el propósito de ser <unk> , comúnmente en China o otros países asiáticos . | ella cree que alrededor de 40 % o más plásticos recolectados en el objetivo de ser reciclados son exportados , comunmente en China o otros países asiáticos . |

| | |
|---|---|
| **ou dans d'autres pays asiatiques.** | |

Ejemplo 2:

| SEGMENTO ORIGINAL | SIN BPE | CON BPE |
|---|---|---|
| **GV : À vos yeux quels sont les obstacles à une plus grande utilisation des nouvelles technologies de l'information et de communication (NTIC) dans les processus électoraux en Guinée et dans la sous-région ?** | GV : a tus ojos los obstáculos a una mayor utilización de la tecnología de la información y comunicación ( <unk> ) en los procesos electorales en Guinea y en la <unk> ? | GV : ¿ En qué los obstáculos a una mayor utilización de las nuevas tecnologías de la información y comunicación ( TIC ) en los procesos electorales en Guinea y la subsiña ? |

Ejemplo 3:

| SEGMENTO ORIGINAL | SIN BPE | CON BPE |
|---|---|---|
| **Ce qui serait désastreux pour l'image de Poutine.** | lo que sería <unk> para la imagen de Putin . | esto sería desastrado por la imagen de Putin . |

Se observa claramente la necesidad de llevar a cabo una postedición tanto en la oración meta sin BPE como en la oración meta con BPE, ya sea porque el sistema ha incluido palabras <unk> para las que no encontraba traducción sin BPE (*collectés*, *exportés*, *NTIC*, *sous-région* y *désastreux*), porque se ha inventado otras con

BPE (*subsiña* y *desatrado*), porque no ha respetado la ortografía de la lengua de llegada (*comunmente*), etc.

## III. Uso del sistema de traducción automática neuronal

Esta parte de la guía describe pasos de preproceso y postproceso que se hacen manualmente en el terminal. Por supuesto, se podrían automatizar en forma de guiones de órdenes del *Shell* o con un programa en Python del estilo de prepare.py usado más arriba, el cual realizaba un preproceso similar previo al entrenamiento.

A la hora de traducir un nuevo texto que proporcione al sistema, deberá seguir los siguientes pasos:

3. Diríjase al directorio DATA o DATABPE, según quiera utilizar un sistema entrenado u otro (con BPE o sin BPE); ambos se encuentran en la carpeta sistema; y sitúe en este directorio un archivo (llamado prueba2.fr) creado a través de un editor de texto sin formato y de forma simple, como por ejemplo, Notepad++, con el siguiente texto:

> Youtubeurs, journalistes et politiciens unissent leurs forces contre les assassinats extrajudiciaires de leaders sociaux (ou défenseurs sociaux, selon un autre terme, NdT) en Colombie.
>
> Avec la campagne #UnChefAMaPlace, ils ouvrent leurs espaces médiatiques pour permettre aux leaders et défenseurs de droits menacés d'écrire ou d'exprimer leurs opinions.
>
> La campagne a démarré le 13 mai, lorsque le journaliste Daniel Samper Ospina a posté sur sa chaîne YouTube un clip d'une chanson de reggaeton avec des dirigeants sociaux et de célèbres YouTubeurs.

   Este texto procede de una publicación del 26 de mayo de 2019 de la edición francesa de *GlobalVoices*[20] en el que se ha colocado cada oración en

[20] https://fr.globalvoices.org/

una línea. La segmentación en línea también se puede realizar automáticamente a través de un programa.

4. Haga el proceso de *tokenización*[21] usando el *tokenizador* del programa *sacremoses* que se ha instalado anteriormente. Para ello, acceda a DATA (o DATABPE si se quiere utilizar BPE) y lance la orden `sacremoses tokenize -l fr <prueba2.fr >prueba2.tok.fr`. Para que sacremoses actúe sobre el fichero que acabamos de crear y cree otro (prueba2.tok.fr) con los resultados de la *tokenización*. Este último fichero ahora tendrá esta forma:

   > Youtubeurs , journalistes et politiciens unissent leurs forces contre les assassinats extrajudiciaires de leaders sociaux ( ou défenseurs sociaux , selon un autre terme , NdT ) en Colombie .
   > Avec la campagne # UnChefAMaPlace , ils ouvrent leurs espaces médiatiques pour permettre aux leaders et défenseurs de droits menacés d' écrire ou d' exprimer leurs opinions .
   > La campagne a démarré le 13 mai , lorsque le journaliste Daniel Samper Ospina a posté sur sa chaîne YouTube un clip d' une chanson de reggaeton avec des dirigeants sociaux et de célèbres YouTubeurs .

   Puede ver que la *tokenización* ha consistido en separar las palabras de los signos de puntuación a los que estaban unidas y que aquellas que tenían apóstrofe han sido separadas igualmente. La *tokenización* reduce efectivamente el tamaño del vocabulario y facilita el aprendizaje.

5. El siguiente paso consiste en pasar a mayúsculas y minúsculas naturales, es decir, el proceso de *truecasing*[22] con la siguiente orden `sacremoses truecase -m truecasemodel.fr <prue-`

---

21 Diríjase al Apéndice I – Glosario para una definición de *tokenización*.
22 Diríjase al Apéndice I – Glosario para la definición de *truecasing*.

`ba2.tok.fr >prueba2.tok.true.fr`. Este último fichero que ha creado sacremoses contiene el siguiente texto:

> youtubeurs , journalistes et politiciens unissent leurs forces contre les assassinats extrajudiciaires de leaders sociaux ( ou défenseurs sociaux , selon un autre terme , NdT ) en Colombie .
> avec la campagne # UnChefAMaPlace , ils ouvrent leurs espaces médiatiques pour permettre aux leaders et défenseurs de droits menacés d' écrire ou d' exprimer leurs opinions .
> la campagne a démarré le 13 mai , lorsque le journaliste Daniel Samper Ospina a posté sur sa chaîne YouTube un clip d' une chanson de reggaeton avec des dirigeants sociaux et de célèbres YouTubeurs .

Se aprecia claramente que palabras como YouTube, *Colombie*, Daniel o el *hashtag* (#) quedan en mayúscula inicial, y mayúscula interna en YouTube o YouTubers, mientras que palabras como *terme, campagne, avec* o *la* permanecen en minúscula. Hay palabras que aún quedan incoherentes como es el caso de YouTubeurs et youtubeurs, porque son tratadas como diferentes por en el modelo de *truecasing*. Se ha usado el modelo truecase-model.fr que se ha generado durante la ejecución de prepare.py en la fase II. Entrenamiento del sistema de traducción automática neuronal.

6. Si se encuentra en DATABPE y quiere partir las palabras en unidades BPE, haga: `subword-nmt apply-bpe --input prueba2.tok.true.fr --codes bpecodes.fr-es --output prueba2.tok.true.bpe.fr`. Esta orden, usando el modelo bpecodes.fr-es generado por prepare.py, produce el fichero prueba2.tok.true.bpe.fr que contiene el siguiente texto:

> y@@ ou@@ tu@@ b@@ eurs , journalistes et politiciens un@@ issent leurs forces contre les assassin@@ ats extr@@ aj@@ u@@ dici@@

aires de leaders sociaux ( ou défenseurs sociaux , selon un autre terme , N@@ d@@ T ) en Colombie .

avec la campagne # Un@@ Ch@@ ef@@ A@@ Ma@@ Pl@@ ace , ils ouv@@ rent leurs espa@@ ces médi@@ atiques pour permettre aux leaders et défenseurs de droits menac@@ és d' écrire ou d' exprimer leurs opinions .

la campagne a démar@@ ré le 13 mai , lorsque le journaliste Daniel Sam@@ per Os@@ p@@ ina a posté sur sa chaîne YouTube un cli@@ p d' une chanson de reg@@ ga@@ et@@ on avec des dirigeants sociaux et de célè@@ bres You@@ Tu@@ b@@ eurs .

Algunas palabras frecuentes se mantienen completas sin haber sido segmentadas por el BPE como *avec*, *dirigeants*, *leaders* o *permettre*. Sin embargo otras menos frecuentes se segmentan como *unissent* (un@@ issent), *assassinats* (assassin@@ ats) o *extrajudiciaires* (extr@@ aj@@ u@@ dici@@ aires).

7. Si quiere ejecutar su sistema de traducción sin BPE sobre este texto, haga la siguiente orden:[23]

```
python -m nmt.nmt ^
    --out_dir=C:\Users\clara\sistema\MODEL ^
    --
    inferen-
    ce_input_file=C:\Users\clara\sistema\DATA\prueba2.tok.true.fr ^
    --inference_output_file= C:\Users\clara\sistema\DATA\salida.es
```

El fichero salida.es contiene la traducción de las tres líneas anteriores, las cuales serán:

[23] Debe tener en cuenta que en los tres parámetros de esta orden se encuentran las localizaciones de mi sistema (el parámetro de out_dir, el de inference_input_file y el de out_dir, por ejemplo, `C:\Users\clara\sistema\DATA\prueba2.tok.true.fr`) y puede que en su máquina no sea la misma, verifíquelas antes de lanzar el entrenamiento.

<unk> , periodistas y políticos <unk> sus fuerzas contra los asesinatos <unk> de líderes sociales ( o defensores sociales ) , según otro término , <unk> ) en Colombia .

con la campaña # <unk> , <unk> sus espacios de medios para permitir a los líderes y defensores de derechos humanos que <unk> o expresar sus opiniones .

la campaña comenzó el 13 de mayo cuando el periodista Daniel <unk> <unk> publicó en su canal YouTube un video de una canción de <unk> con líderes sociales y famosos <unk> .

En cada texto es necesario restituir las mayúsculas correctas y después juntar las palabras. Esto se puede hacer de golpe, juntando las dos órdenes en una tubería con "|". Desde DATA haga `sacremoses detruecase <salida.tok.true.es | sacremoses detokenize >salida.es`. El resultado es el siguiente:

<unk>, periodistas y políticos <unk> sus fuerzas contra los asesinatos <unk> de líderes sociales (o defensores sociales), según otro término, <unk>) en Colombia.

Con la campaña # <unk>, <unk> sus espacios de medios para permitir a los líderes y defensores de derechos humanos que <unk> o expresar sus opiniones.

La campaña comenzó el 13 de mayo cuando el periodista Daniel <unk> <unk> publicó en su canal YouTube un video de una canción de <unk> con líderes sociales y famosos <unk>.

5. Si quiere ejecutar su sistema de traducción con BPE sobre este texto, haga la siguiente orden:[24]

```
python -m nmt.nmt ^
      --out_dir=C:\Users\clara\sistema\MODELBPE ^
```

[24] Al igual que en la note a pie de página anterior, las localizaciones son ejemplos de mi sistema, verifique los suyos antes de lanzar el entrenamiento.

```
--subword_option=bpe ^
--
inferen-
ce_input_file=C:\Users\clara\sistema\DATABPE\prueba2.tok.true.fr
^
--
inferen-
ce_output_file=C:\Users\clara\sistema\DATABPE\salida.tok.true.es
```

El fichero salida.bpe.es contiene las tres líneas traducidas:

> los periodistas y políticos unan a sus fuerzas contra los asesinatos extrajudiciarios de líderes sociales ( o defensores sociales , según un nuevo término , NdT ) en Colombia .
> con la campaña # UnChefAMaPlace , abran sus espacios mediáticos para permitir a los líderes y defensores de derechos de escribir o expresar sus opiniones .
> la campaña comenzó el 13 de mayo cuando el periodista Daniel Samper Ospina publicó en su canal de YouTube un video de una canción de reggaeton con líderes sociales y famosos YouTubadores .

Como se ha explicado en el apartado anterior (5. A.) es necesario poner mayúsculas naturales y después juntar las palabras. Por lo que desde DATABPE haga `sacremoses detruecase <salida.tok.true.es | sacremoses detokenize >salida.es`, obteniendo el siguiente resultado:

> Los periodistas y políticos unan a sus fuerzas contra los asesinatos extrajudiciarios de líderes sociales (o defensores sociales, según un nuevo término, NdT) en Colombia.
> Con la campaña #UnChefAMaPlace, abran sus espacios mediáticos para permitir a los líderes y defensores de derechos de escribir o expresar sus opiniones.

> La campaña comenzó el 13 de mayo cuando el periodista Daniel Samper Ospina publicó en su canal de YouTube un video de una canción de reggaeton con líderes sociales y famosos YouTubadores.

En este ejemplo se ve claramente que el resultado con BPE funciona mejor.

En estos ejemplos se ha introducido un texto pequeño de tres líneas para comprobar cómo funcionaba el sistema de traducción automática neuronal que ha creado. Si el texto fuera más largo, como sería en el caso de un encargo de traducción típico, se podría hacer el mismo proceso, a partir del paso 5. A. (opción sin BPE) o 5. B. (opción con BPE) en la fase III. Uso del sistema de traducción automática neuronal. Al final, se podría crear una memoria de traducción automática en formato TMX a partir del archivo original (en este ejemplo, prueba2.fr) y del archivo traducido (en este ejemplo, salida.es) usando, por ejemplo, el alineador incorporado en OmegaT a partir de la versión 4 o cualquier otro alineador de textos. Después, esta memoria TMX se pondría en un proyecto con el texto prueba2.fr y entonces solo habría que posteditar el resultado para que el resultado cumpla el encargo de traducción.

# Apéndice II – Glosario

La finalidad de este glosario es aclarar las posibles dudas sobre conceptos relacionados con las tecnologías de la traducción con sus equivalentes ingleses que le podrían surgir a una persona ajena a este campo para llevar a cabo la elaboración de un traductor automático neuronal.

Aprendizaje profundo (*deep learning*): es una subcategoría del aprendizaje automático, el aprendizaje profundo trata del uso de redes neuronales para mejorar cosas tales como el reconocimiento de voz, la visión por ordenador y el procesamiento del lenguaje natural. Se denomina «profundo» porque hace referencia al uso de numerosas capas ocultas en redes neuronales artificiales.

Fuente: https://www.bbvaopenmind.com/que-es-el-aprendijaze-profundo/

Archivo .bat (*batch file*): es un archivo *batch*, es decir, un archivo de procesamiento por lotes. Se trata de archivos de texto sin formato, guardados con la extensión .BAT que contienen un conjunto de instrucciones para Windows. Cuando se ejecuta este archivo, las órdenes contenidas son ejecutadas en grupo, de forma secuencial, lo que permite automatizar diversas tareas, o formando bucles, saltándose partes.

Fuente: https://es.wikipedia.org/wiki/Archivo_batch

Bash (*Bourne-again Shell*): es un programa informático, cuya función consiste en interpretar órdenes, y el lenguaje de consola asociado. Es un intérprete de comandos de Unix compatible con POSIX; de hecho, es el intérprete de comandos que se ejecuta por defecto en la mayoría de las distribuciones GNU/Linux, además de en MacOS. También se ha llevado a otros sistemas como Windows y Android.

Fuente: https://es.wikipedia.org/wiki/Bash

BLEU (*BiLingual Evaluation Understudy*): es un indicador de evaluación automático de la calidad de traducciones realizadas por sistemas de TA. Una traducción tiene mayor valor de BLEU cuanto más similar es con respecto a otra de referencia, que se supone correcta. BLEU compara cuántas secuencias de una, dos, tres y cuatro palabras coinciden entre la traducción automática y la de referencia, y agrupa las cuatro cuentas en un único indicador. Una desventaja es que no les da valor a las propuestas cercanas, es decir, a los sinónimos, ya que valora sólo las coincidencias exactas. Su rango es de 0 a 1, o de 0 % a 100 %.

Fuente: https://es.wikipedia.org/wiki/BLEU

BPE: abreviatura de *byte pair encoding*, es un método de compresión de textos que primero asigna un código a cada carácter (aproximadamente un *byte*), y después agrupa de dos en dos (*pair*) las secuencias de caracteres más comunes asignándoles un código único, hasta que se llega a un número determinado de códigos. Se usa en traducción automática basada en corpus para poder descomponer las palabras en unidades que son susceptibles de repetirse más a menudo que las palabras completas, y así mitigar el problema que comporta traducir palabras que no se han visto durante el entrenamiento.

Fuente: Mikel L. Forcada, comentario privado.

Capa (*layer*): es un grupo de neuronas que recibe conexiones de neuronas de fuera de la capa y las envía a neuronas de fuera de la capa. Un traductor automático neuronal trabaja con múltiples capas de neuronas, es decir, las de entrada, las de salida y las que están ocultas (→ *aprendizaje profundo*), que por lo tanto no se muestran y conectan las de entrada con las de salida.

Fuente: Mikel L. Forcada, comentario privado.

Codificador (*encoder*): es una red neuronal (normalmente recurrente) que analiza de izquierda a derecha y de derecha a izquierda la frase origen palabra por palabra para producir una representación vectorial de la misma.

Fuente: Casacuberta Nolla, F. & Peris Abril, A. (2017) "Traducción automática neuronal." *Revista Tradumàtica. Tecnologies de la Traducció 15*, pp. 66-74 Versión electrónica: <https://ddd.uab.cat/pub/tradumatica/tradumatica_a2017n15/tradumatica_a2017n15p66.pdf>.

Descodificador (*decoder*): es una red neuronal (normalmente recurrente) que, en un instante dado, genera una palabra destino tomando como entrada la palabra previamente generada, el estado de la red neuronal en el instante anterior y una representación de la frase origen a partir de la información suministrada por el codificador.

Fuente: Casacuberta Nolla, F. & Peris Abril, A. (2017) "Traducción automática neuronal." *Revista Tradumàtica. Tecnologies de la Traducció 15*, pp. 66-74 Versión electrónica: <https://ddd.uab.cat/pub/tradumatica/tradumatica_a2017n15/tradumatica_a2017n15p66.pdf>.

Desarrollo (*development* o *validation set*): es un corpus formado por entre 1 000 y 3 000 pares de frases que determinan al sistema cuándo dejar el entrenamiento para no «sobreentrenarlo». Junto con el corpus de entrenamiento y el de test forman los tres corpus necesarios para el entrenamiento de un sistema de TAN.

Fuente: https://www.slideshare.net/mlforcada/cairo-2019seminar

Entorno virtual de Python (*virtual environment*): entorno (gestionado por un gestor de entornos virtuales tal como VirtualEnv, Anaconda, etc) en el que se pueden, en el mismo equipo, desarrollar proyectos o usar programas en Python

que usan diferentes versiones de paquetes que, de otra forma, entrarían en conflicto.

Fuente: https://wiki.archlinux.org/index.php/Python/Virtual_environment_(Espa%C3%B1ol)

Entrenamiento (*training set*): es un corpus formado por entre 100 000 y 1 000 000 pares de frases que sirven al sistema para entrenarlo con sus pares de frases alineados. Junto con el corpus de desarrollo y el de test forman los tres corpus necesarios para el entrenamiento de un sistema de TAN.

Fuente: https://www.slideshare.net/mlforcada/cairo-2019seminar

Git Bash: no es sólo un *bash* compilado para Windows que se instala juntamente con el gestor de versiones Git, sino un paquete más completo que contiene *bash* (un intérprete de línea de órdenes) y una recopilación de otras utilidades que se ejecutan usando el intérprete, compiladas para Windows, y una nueva ventana de terminal de interfaz de línea de comandos llamada *mintty*.

Fuente: definición propia.

GitHub: es una plataforma de desarrollo colaborativo para alojar proyectos que usa, entre otros, el sistema de control de versiones Git, similar a Gitlab. Se utiliza principalmente para la creación, el almacenamiento y la gestión de código fuente de programas de ordenador. Además de gestor de repositorios, el servicio ofrece también alojamiento de wikis y un sistema de seguimiento de errores.

Fuente: https://es.wikipedia.org/wiki/GitHub

Gitlab: es una plataforma de código abierto de control de versiones y desarrollo de software colaborativo basado en Git, similar a GitHub Además de gestor de

repositorios, el servicio ofrece también alojamiento de wikis y un sistema de seguimiento de errores.

Fuente: https://es.wikipedia.org/wiki/GitLab

GNU/Linux: es un sistema operativo libre del tipo de Unix, multiplataforma, multiusuario y multitarea. Todo su código fuente puede ser utilizado, modificado y redistribuido libremente por cualquiera, normalmente bajo los términos de la GPL (Licencia Pública General de GNU).

Fuente: https://es.wikipedia.org/wiki/GNU/Linux

Inteligencia artificial (*artificial intelligence*): es una disciplina científica que se ocupa de crear programas informáticos que ejecutan operaciones comparables a las que realiza la mente humana, como el aprendizaje o el razonamiento lógico.

Fuente: https://dle.rae.es/?id=LqtyoaQ|LqusWqH

Intérprete de órdenes (*command prompt*): es un terminal o consola donde las órdenes se teclean textualmente en un lenguaje determinado que sabe interpretar.

Fuente: Mikel L. Forcada, comentario privado.

Lengua controlada (*controlled language*): es un subconjunto de un lenguaje natural obtenido mediante la restricción de su gramática y vocabulario para que sea más fácil de procesar por los ordenadores, por ejemplo, eliminando ambigüedades y otras características del lenguaje natural que dificultan la TA.

Fuente: Forcada, M. L.; Pérez, L. & Rico, C. (2018) *Translators' track: a glossary*. Alicante: EAMT. Versión electrónica: <http://eamt.org/translators_documents/eamt_2018_glossary.pdf>.

Microsoft Windows: es el nombre de una familia de distribuciones de software para PC, smartphone, servidores y sistemas empotrados, desarrollados y

vendidos por Microsoft y disponibles para múltiples arquitecturas, tales como x86, x86-64 y ARM. No son sistemas operativos, sino que contienen uno (tradicionalmente MS-DOS, o el más actual cuyo núcleo es Windows NT) junto con una amplia variedad de software.

Fuente: https://es.wikipedia.org/wiki/Microsoft_Windows

Modelo de lengua (*language model*): modelo matemático asigna una probabilidad o calificación a una secuencia de palabras, generalmente a una frase. Normalmente se utilizan en la traducción automática estadística para crear traducciones más fluidas.

Fuente: Forcada, M. L.; Pérez, L. & Rico, C. (2018) *Translators' track: a glossary*. Alicante: EAMT. Versión electrónica: <http://eamt.org/translators_documents/eamt_2018_glossary.pdf>.

Mecanismo de atención (*attention model*): es una subred neuronal, que se añade a una arquitectura de traducción automática neuronal codificador–descodificador, que permite al descodificador «prestar» atención a determinadas posiciones en las representaciones correspondientes a la frase original.

Fuente: Casacuberta Nolla, F. & Peris Abril, A. (2017) "Traducción automática neuronal." *Revista Tradumàtica. Tecnologies de la Traducció 15*, pp. 66-74 Versión electrónica: <https://ddd.uab.cat/pub/tradumatica/tradumatica_a2017n15/tradumatica_a2017n15p66.pdf>.

Neurona artificial (*artificial neuron*): es una unidad de procesamiento vagamente inspirada en una neurona natural, es una unidad computacional muy simple. Su estado o activación depende de los estímulos recibidos de otras neuronas artificiales (o de entradas) a través de sus conexiones.

Fuente: Forcada, M. L.; Pérez, L. & Rico, C. (2018) *Translators' track: a glossary*. Alicante: EAMT. Versión electrónica: <http://eamt.org/translators_documents/eamt_2018_glossary.pdf>.

Notepad++: es un editor de texto libre y de código fuente abierto con soporte para varios lenguajes de programación. Se parece al Bloc de notas en cuanto al hecho de que puede editar texto sin formato y de forma simple. No obstante, incluye opciones más avanzadas para desarrolladores y programadores.

Fuente: https://es.wikipedia.org/wiki/Notepad%2B%2B

Iteraciones (*steps*): en un algoritmo de entrenamiento de una red neuronal se corresponde con la presentación de un lote de ejemplos a la misma y con una actualización de los pesos para aprenderlos. Por ejemplo, a un sistema de traducción automática neuronal se le pueden presentar 128 ejemplos en cada paso o iteración.

Fuente: Mikel L. Forcada, comentario privado.

Pip (*pip*): es un gestor de paquetes que se utiliza para instalar desde Internet y administrar los paquetes del software de Python.

Fuente: https://es.wikipedia.org/wiki/Pip_(administrador_de_paquetes)

Python: es un lenguaje de programación multiparadigma. Esto significa que más que forzar a los programadores a adoptar un estilo particular de programación, permite varios estilos: programación orientada a objetos, programación imperativa y programación funcional. Tensorflow utiliza pequeños programas denominados *scripts* que tienen que ser interpretados por Python.

Fuente: https://es.wikipedia.org/wiki/Python

Red neuronal artificial (*artificial neural network*): es un conjunto de neuronas artificiales conectadas de una manera específica para realizar una tarea computacional determinada. Las conexiones tienen una fuerza o un peso que se puede aprender y que regula cómo se ve afectado el estado o la acti-

vación de una neurona artificial por la activación de las neuronas vecinas (o por el valor de las entradas), para que toda la red tenga el comportamiento que se desea.

Fuente: Forcada, M. L.; Pérez, L. & Rico, C. (2018) *Translators' track: a glossary*. Alicante: EAMT. Versión electrónica: <http://eamt.org/translators_documents/eamt_2018_glossary.pdf>.

*Word embedding*: es la representación vectorial de una palabra, usada en la traducción automática neuronal y a su vez es normalmente la activación de una capa de neuronas. Los *word embeddings* se aprenden generalmente a partir de grandes textos monolingües. Para palabras similares, sus representaciones son matemáticamente parecidas. Por ejemplo, el vector (0.35, 0.28, –0.15, 0.76, …, 0.88) puede ser la representación de la palabra «study».

Fuente: Forcada, M. L.; Pérez, L. & Rico, C. (2018) *Translators' track: a glossary*. Alicante: EAMT. Versión electrónica: <http://eamt.org/translators_documents/eamt_2018_glossary.pdf>.

Tamaño del lote (*batch size*): es un término utilizado en el aprendizaje automático y se refiere al número de ejemplos de entrenamiento utilizados antes del cálculo de cada paso o iteración, o sea, de una actualización de los pesos.

Fuente: https://radiopaedia.org/articles/batch-size-machine-learning

TensorFlow: es una *librería* (un anglicismo, de *library*, biblioteca de programas) de código abierto, es decir, se puede acceder a ella libremente y un programador puede modificarla según las necesidades del usuario. Tensorflow permite construir, entrenar y usar redes neuronales artificiales. Este sistema es un codificador–descodificador, una posible arquitectura para los sistemas de TAN. Tensorflow utiliza pequeños programas denominados scripts que tienen que ser interpretados por Python.

Fuente: https://es.wikipedia.org/wiki/TensorFlow

Test (*set test*): es un corpus formado por entre 1 000 y 3 000 pares de frases que proporcionan una idea del rendimiento del sistema de traducción automática neuronal. Junto con el corpus de entrenamiento y el de desarrollo forman los tres corpus necesarios para el entrenamiento de un sistema de TAN.

Fuente: https://www.slideshare.net/mlforcada/cairo-2019seminar

*Tokenización* o segmentación (*tokenization*): es la segmentación de texto original en *tokens*, generalmente (pero no siempre) en palabras o unidades similares, antes de enviarlo a un sistema de TA: No iremos a la fiesta de Ben → We + wo + n't + go + to + Ben + 's + party. Esta tokenización debe deshacerse antes de obtener el TM.

Fuente: Forcada, M. L.; Pérez, L. & Rico, C. (2018) *Translators' track: a glossary*. Alicante: EAMT. Versión electrónica: <http://eamt.org/translators_documents/eamt_2018_glossary.pdf>.

Traducción automática basada en corpus (*corpus-based machine translation*): es un tipo de TA que utiliza el conocimiento aprendido de corpus paralelos de miles o millones de pares de frases que contienen segmentos de la LO y sus traducciones a la LM. Como, por ejemplo, la traducción automática estadística y la neuronal.

Fuente: Forcada, M. L.; Pérez, L. & Rico, C. (2018) *Translators' track: a glossary*. Alicante: EAMT. Versión electrónica: <http://eamt.org/translators_documents/eamt_2018_glossary.pdf>.

Traducción automática basada en reglas (*rule-based machine translation*): es un tipo de TA que funciona aplicando reglas gramaticales y diccionarios escritos por expertos, en contraste con lo que hace la basada en corpus.

Fuente: Forcada, M. L.; Pérez, L. & Rico, C. (2018) *Translators' track: a glossary*. Alicante: EAMT. Versión electrónica: <http://eamt.org/translators_documents/eamt_2018_glossary.pdf>.

Traducción automática estadística, TAE (*statistical machine translation*): es un tipo de TA basada en corpus que genera traducciones usando modelos probabilísticos que a su vez se obtienen realizando estadísticas sobre los corpus paralelos. Estos sistemas asignan una probabilidad aproximada a muchas traducciones posibles del TO y luego eligen la más probable.

Fuente: Forcada, M. L.; Pérez, L. & Rico, C. (2018) *Translators' track: a glossary*. Alicante: EAMT. Versión electrónica: <http://eamt.org/translators_documents/eamt_2018_glossary.pdf>.

Traducción automática neuronal, TAN (*neural machine translation*): es un tipo de TA basada en corpus que utiliza redes neuronales artificiales entrenadas sobre corpus paralelos de miles o millones de pares de frases. Una gran parte de los sistemas de TAN consisten en un codificador y un descodificador, cada uno de los cuales es una red neuronal (recurrente) artificial especializada.

Fuente: Forcada, M. L.; Pérez, L. & Rico, C. (2018) *Translators' track: a glossary*. Alicante: EAMT. Versión electrónica: <http://eamt.org/translators_documents/eamt_2018_glossary.pdf>.

Traducción automática basada en frases (*phrase-based machine translation*): es un tipo de TAE que usa pares de frases, es decir, correspondencias entre las secuencias de una o más palabras contiguas. No tienen que ser necesariamente frases en el sentido lingüístico de la palabra.

Fuente: Forcada, M. L.; Pérez, L. & Rico, C. (2018) *Translators' track: a glossary*. Alicante: EAMT. Versión electrónica: <http://eamt.org/translators_documents/eamt_2018_glossary.pdf>.

Traducción basada en palabras (*word-based machine translation*): es un tipo de TAE en la que la unidad esencial de la traducción es una palabra, en contraste con la →*Traducción automática basada en frases*.

Fuente: https://es.wikipedia.org/wiki/Traducci%C3%B3n_autom%C3%A1tica_estad%C3%ADstica#Traducci%C3%B3n_basada_en_palabras_(Word-Based_translation)

Traducción neuronal automática con atención (*attention-based machine translation*): es un tipo de TAN que aprende a usar una técnica llamada *atención* que se añade a una arquitectura de traducción automática neuronal codificador–descodificador y que permite al descodificador «prestar» atención a determinadas posiciones en las representaciones correspondientes a la frase original.

Fuente: Forcada, M. L.; Pérez, L. & Rico, C. (2018) *Translators' track: a glossary*. Alicante: EAMT. Versión electrónica: <http://eamt.org/translators_documents/eamt_2018_glossary.pdf>.

*Truecasing* (de *true*, «verdadero» y *case*, «caja» de impresor en el sentido de mayúsculas –caja alta– o minúsculas –caja baja–): es un método en el procesamiento de lenguajes naturales que sirve para reconstruir de manera correcta (*true*) las mayúsculas de las palabras cuando en el corpus de entrenamiento estas se han presentado siempre en su forma más frecuente.

Fuente: Mikel L. Forcada, comentario privado.

UTF-8: es un formato de codificación de caracteres Unicode que usa símbolos de longitud variable de 1 a 6 bytes. Actualmente es una de las tres posibilidades de codificación reconocidas por Unicode y lenguajes web. Divide los

caracteres Unicode en varios grupos, en función del número de bytes necesarios para codificarlos.

Fuente: https://es.wikipedia.org/wiki/UTF-8

Vector (*vector*): es una lista ordenada y de tamaño fijo de números reales como «(-0.73. 0.31, 0.76, -0.02)».

Fuente: Forcada, M. L.; Pérez, L. & Rico, C. (2018) *Translators' track: a glossary*. Alicante: EAMT. Versión electrónica: <http://eamt.org/translators_documents/eamt_2018_glossary.pdf>.

# Apéndice III – Órdenes para Bash y para *Shell* de Windows

En este apéndice, se reúnen algunas órdenes tanto de Bash como del *Shell* de Windows que se utilizan a lo largo de la guía y cuyo conocimiento será más que útil. Como podrá ver, algunas órdenes se repiten en ambos apartados puesto que las comparten los dos intérpretes de órdenes y otras no.

## I. Bash

`|` → tubería (*pipeline*) o cadena de montaje, conectar los comandos para unir la salida de uno con la entrada del siguiente.

`c < X` → dirigir el texto del archivo (X) a la entrada estándar de la orden C (si no se pone `< X`, se toma del teclado).

`c > X` → dirigir la salida estándar de la orden C al fichero (X) (si no se pone > X, se imprime en la pantalla).

`cat > X` → leer de la entrada estándar y escribir la salida estándar en el fichero (X) borrando lo que había inicialmente en el fichero.

`cat X` → envía el fichero (X) a la salida estándar (normalmente la pantalla del intérprete).

`cat X Y` → concatenar y enviar los ficheros a la salida estándar (normalmente la pantalla del intérprete).

`cd` → dirigirse al directorio principal.

`cd X` → ir al directorio (X).

`cp X Y`→ copiar un archivo de un fichero (X) a otro (Y), por ejemplo: `cp C:/Users/clara/prueba/invisible.docx C:/Users/clara/pruebaGPU`.

`date` → mostrar la fecha.

`grep` → tomar una expresión regular del intérprete, leer la entrada estándar o una lista de archivos, e imprimir las líneas que contengan coincidencias para la expresión. Por ejemplo:

`grep "^c.*ma$"` → con .* enseña todas las líneas que empiecen por «c», y terminen en «ma», teniendo cualquier letra o letras entre medias (*cúrcuma*, *coma*, *crema*, etc).

`grep "^X$"` → para buscar una línea específica (X) en archivos del TAN, siendo ^ el inicio de línea y $ el final de esta.

`help` → mostrar una lista con todos los comandos disponibles.

`less X` → desplegar el archivo de texto X de manera que se puede navegar en él, buscar, etc. (usando, por ejemplo, la tecla «intro» para avanzar, tecla «q»). Cuando el archivo esté abierto, pulsando la tecla «h», proporciona ayuda.

`ls *` → mostrar todos los programas y archivos del directorio y lo que contienen.

`ls *.odt` → mostrar todos los archivos .odt del directorio. Se podría hacer con otra extensión de archivos, esta es solo un ejemplo (también se podría `ls *.pdf`).

`ls` → mostrar los archivos que tiene el directorio actual.

`mv X Y` → mover un archivo (X) a otro directorio (Y) que esté dentro del mismo directorio en el que se encuentra o renombrar un archivo como Y. También sirve para cambiar el nombre de un directorio o archivo (X) a otro (Y).

`rm *.*` → eliminar todos los ficheros dentro del directorio actual.

`rm X` → eliminar un archivo (X).

`sort -r -u X` → ordenar el contenido del archivo (X) en sentido inverso alfabéticamente y omitiendo las palabras repetidas.

`sort -r X` → ordenar el contenido del archivo (X) en sentido inverso alfabéticamente.

`sort X` → ordenar el contenido del archivo (X) alfabéticamente.

`wc -w X` → contar cuántas palabras tiene un archivo (X).

`X -V` o `x -v` (depende del programa) → mostrar la versión del programa en cuestión (X).

`X --version` → mostrar la versión del programa en cuestión (X).

Con algunos de estos comandos, podríamos crear múltiples combinaciones:

`cat ../data/train.tok.true.es | tr ' ' '\012' | grep "^cama$" | wc -l` → nos mostrará el número de veces que aparece la palabra «cama»;

`cat ../data/train.tok.true.es | tr ' ' '\012' | grep "^c.*ma$" | sort | uniq` → nos mostrará alfabéticamente todas las palabras que empiecen por «c» y acaben en «ma» sin repeticiones;

`paste ../data/prueba.fr ../data/prueba.es` → pegar línea a línea, lado a lado el contenido de los dos archivos. En este caso, un archivo en lengua francesa con su correspondiente traducción automática al español.

### II. *Shell* de Windows

`cd ..` → desplazarse al directorio anterior (padre).

`cd` → mostrar el nombre del directorio actual.

`cd X` → ir al directorio (X).

cmd → en el buscador del ordenador, ejecutar el símbolo del sistema, también llamado línea de comandos.

`copy X Y` → copiar un archivo de un fichero (X) a otro (Y), por ejemplo: `copy C:\Users\clara\prueba\invisible.docx C:\Users\clara\pruebaGPU`.

`date` → mostrar la fecha y cambiarla si lo desea.

`del *.*` o `erase *.*` → eliminar todos los ficheros dentro del directorio actual.

`del .` o `erase .` → eliminar todos los ficheros dentro del directorio actual.

`del X` o `erase X` → eliminar un archivo o directorio (X).

`dir *.odt` → mostrar todos los archivos .odt del directorio. Se podría hacer con otra extensión de archivos, esta es solo un ejemplo (también se podría dir *.pdf).

`dir` → mostrar los archivos que tiene el directorio actual.

`exit` → salir del símbolo del sistema.

`find /c /v "" X` → contar cuántas palabras tiene un archivo (X) (usando una configuración la orden «find» que busca líneas en un archivo que cumplan ciertas condiciones, parecida a «grep» del *shell* bash).

`help` → mostrar una lista con todos los comandos disponibles.

`help X` → mostrar información sobre un comando específico (X).

`md X` o `mkdir X` → crear un directorio (X).

`mkdir X` → crear un directorio (X).

`more X` → presentar en la pantalla el archivo (X) pantalla a pantalla (usando la tecla «intro» para avanzar).

`move X Y` → mover un archivo (X) a otro directorio (Y) que esté dentro del mismo directorio en el que se encuentra o renombrar un archivo como Y.

`ren X Y` o `rename X Y` → renombrar un archivo (X) como Y.

`rmdir /s X` → eliminar todos los directorios y archivos del directorio (X) además del mismo directorio.

`rd X` → eliminar el directorio (X) si este está vacío.

`sort X` → ordenar el contenido del archivo (X) alfabéticamente.

`time` → mostrar la hora y cambiarla si lo desea.

`ver` → mostrar la versión de Windows.

`X /?` → mostrar todos los modificadores y posibilidades del comando (X).

`X -V` o `X -v` → mostrar la versión del programa en cuestión (X).

`X --version` → mostrar la versión del programa en cuestión (X).

### III. Git

`git clone X` → *clonar* (crear una copia local) el repositorio que se encuentra en la URL (por ejemplo, `git clone https://gitlab.com/mlforcada/corpus-utils/` clona el repositorio *corpus-utils* del usuario mlforcada alojado en la plataforma gitlab). Sirve para descargarse la última versión en desarrollo de un programa, aunque no vayamos a modificarlo.

`git pull` → actualizar el repositorio local (clon del remoto) para que refleje los cambios realizados en el repositorio remoto.

`git push` → actualizar el repositorio remoto los cambios comprometidos con 'git commit' en el repositorio local del cual es clon.

`git commit` → comprometer (registrar como cambios en el repositorio local) los cambios realizados en ficheros del repositorio local.

Estos tres últimos comandos no se usan en la guía, pero podrían ser útiles posteriormente.

### IV. Acciones de teclado y de ventana

↹ (tecla del tabulador) → completa el nombre tecleado o da sugerencias para completarlo para ser más eficiente a la hora de buscar archivos con nombres largos o complicados.

`control c` → acabar con un comando en ejecución.

control → abrir el Panel de control de la máquina.

↓ / ↑ → mostrar los comandos usados anteriormente.

Arrastrar y soltar archivos → Si se trabaja con archivos en un directorio profundamente anidado y se necesitan sus nombres de ruta completos para un

comando, hay diferentes opciones: escribirlos manualmente, copiar la ruta desde la barra de direcciones del Explorador de archivos y luego escribir el nombre del archivo; o arrastrar y soltar el archivo directamente en el símbolo del sistema.

nombre del archivo → solo en *Shell* de Windows, abrir un archivo dentro del directorio actual (por ejemplo, con «tfg.odt» el cmd abrirá el archivo en cuestión con la aplicación que corresponda).

F1 → escribir el último comando utilizado, a carácter por pulsación.

F3 → escribir completamente el contenido de la última línea de comandos utilizada.

F5 → volver al último comando utilizado.